Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Mihec Podlesek

Ovrednotenje testnega procesa v podjetju in praktičen prikaz testiranja z orodjem SoapUi

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:
Ovrednotenje testnega procesa v podjetju in praktičen prikaz testiranja z orodjem SoapUi

Tematika naloge:

Uspešnost podjetja za razvoj programske opreme je neposredno povezana z uporabo ustreznega procesa razvoja. Nadgradnja le-tega zahteva sistematično presojo, pri kateri nam pomagajo znani modeli za zagotavljanje kakovosti. V diplomski nalogi predstavite strukturo in proces razvoja konkretne razvojne skupine v tovrstnem podjetju in ga ovrednotite v skladu s smernicami zmožnostno zrelostnega modela (CMM). Izpostavite področje testiranja programske opreme, pri čemer prikažite tudi njegovo praktično izvedbo z uporabo testnega orodja SoapUi. Na podlagi ugotovitev ovrednotenja in prikaza predlagajte konkretna priporočila za izboljšanje področja testiranja v podjetju.

Hvala mentorju viš. pred. dr. Igorju Rožancu za vse vzpodbude, nasvete, potrpežljivost ter pomoč pri nastanku diplomske naloge.
Hvala celotni družini še posebej mami in partnerici Barbari za leta nesebične podpore, odrekanja in potrpežljivosti.
Hvala prijateljem, še posebej Primožu in Gorazdu.
Iskrena hvala tudi Janiju, Boru in Sajotu, s katerimi smo skupaj pričeli študijsko pot na FRI, prav tako pa Mateju, Sebastianu, Roku in Boštjanu, s katerimi smo se spoznali malenkost kasneje. Posebna zahvala pa gre mojemu stricu Jožetu, ki me je že v rani mladosti navdušil za to, da stopim v svet računalništva in informatike.
Hvala tudi vsem ostalim, ki ste mi stali ob strani ali kako drugače pripomogli k nastanku diplomske naloge in morebiti niste omenjeni.

Prijateljema.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	Aplikacijski programski vmesnik
CMM	Capability Maturity Model	Model stopenj zrelosti
CMMI	Capability Maturity Model Integration	Sestavljeni model stopenj zrelosti
HTML	Hyper Text Markup Language	Jezik za označevanje nadbesedila
JDBC	Java Database Connectivity	Javanski vmesnik za povezovanje s podatkovnimi zbirkami
JMS	Java Message Service	Prilagodljiva storitev za asinhrono izmenjavo kritičnih poslovnih podatkov in dogodkov
JSON	JavaScript Object Notation	notacija objektov JavaScript
JVM	Java Virtual Machine	Javansko virtualno okolje
PCMM	People Capability Maturity Model	Model stopenj zrelosti upravljanja s človeškimi viri
REST	Representational State Transfer	Predstavitveni prenos stanja
SOAP	Simple Object Access Protocol	Protokol za preprost dostop do objektov
TMMI	Test Maturity Model Integration	Sestavljeni model stopenj zrelosti za proces testiranja
WADL	Web Application Description Language	Opisni jezik spletnih aplikacij, ki slonijo na HTTP protokolu
WSDL	Web Services Description Language	Opisni jezik spletnih storitev
XML	Extensible Markup Language	Razširljivi označevalni jezik

Povzetek

Naslov: Ovrednotenje testnega procesa v podjetju in praktičen prikaz testiranja z orodjem SoapUi

Avtor: Mihec Podlesek

Namen diplomske naloge je predstaviti proces razvoja programske opreme v enem izmed večjih slovenskih IT podjetij s poudarkom na fazi testiranja in ga s pomočjo modela stopenj zrelosti (CMM) objektivno ovrednotiti. Poleg tega želimo predstaviti osnovne funkcionalnosti in značilnosti odprtokodnega testnega orodja SoapUI, ki smo ga uporabljali v procesu testiranja v podjetju. V ta namen prikažemo tudi testiranje, z omenjenim orodjem.
V diplomski nalogi z analizo procesa razvoja in testiranja PO ugotovimo, da se podjetje nahaja na prvi, oziroma začetni fazi modela stopenj zrelosti CMM. S prikazom uporabe orodja SoapUI pa izpostavimo tudi omejitve tega orodja v procesu testiranja. Na podlagi ovrednotenja podamo predloge za izboljšanje procesa testiranja PO, ki jih lahko podjetje neposredno implementira in se tako približa drugi stopnji modela CMM. Ključni ukrepi, ki jih predlagamo so uvedba ločene ekipe za namen testiranja PO, uvedba strokovnega mentorskega programa za novozaposlene in sledenje napakam med procesom razvoja PO ter beleženje le-teh.

Ključne besede: CMM, SCRUM, razvoj PO, testiranje PO.

Abstract

Title: Evaluation of a company's testing process and a practical demostration of testing with SoapUi

Author: Mihec Podlesek

Purpose of this bachelor thesis is to present the process of software development with emphasis on software testing in one of the biggest Slovenian IT company and to do an objective evaluation based on the CMM model. We also present basic functionalities of the software testing tool SoapUI, which we used in the company. Throughout the analysis of the processes of software development and software testing, we find that the company is at the first, or initial level of the CMM model. We also present a few practical test cases with SoapUI tool and discover limitations in regard to its usage in the company's testing process. Based on these findings we then propose solutions, which can be directly implemented by the company. Key solutions that we propose for the company are to divide the analytic/tester role and introduce a new team for the sole purpose of testing computer software in the company, introduction of traineeship programmes for new employees and active tracking and logging of errors which are detected throughout the software development process.

Keywords: CMM, SCRUM, software development, software testing.

Poglavje 1

Uvod

1.1 Motivacija in cilji

Motivacija za izbiro teme diplomskega dela je bila izkušnja med obvezno delovno prakso v enem izmed večjih slovenskih IT podjetij, ki med drugim razvija in vzdržuje tudi portal eUprava [14]. Delo bi lahko razdelili na administrativni in tehnični del. Administrativni del je obsegal pomoč pri projektnemu vodenju, organizacijo sestankov z različnimi deležniki projekta in komunikacijo z naročnikom. Tehnični del je bil usmerjen v raziskovanje novih funkcionalnosti in pripravo funkcionalnosti na implementacijo, testiranje PO in razvoj, ter razvoj testnih scenarijev.
Med sodelovanjem v skupini E2 smo opazili, da obstaja še veliko prostora za izboljšanje procesa razvoja in testiranja PO. Ob sodelovanju v skupini smo sistematično analizirali pristop skupine k omenjenima procesoma in poskušali ugotoviti, na kateri stopnji v modelu CMM se podjetje nahaja in kaj bi bilo potrebno storiti, da se ta dvigne na višji nivo. Cilji diplomske naloge so temu primerni: predstaviti proces razvoja PO v podjetju s poudarkom na fazi testiranja in ga s pomočjo izbrane metodologije objektivno ovrednotiti na podlagi modela stopenj zrelosti CMM. Da bi prikazali praktično plat testiranja, bomo predstavili tudi programsko orodje SoapUI, s katerim se v podjetju izvaja osrednje testiranje PO. To orodje bo predstavljeno na praktičnem pri-

meru. Na koncu smo ugotovitve strnili v priporočila za izboljšanje procesa razvoja PO s poudarkom na fazi testiranja.

1.2 Struktura diplomske naloge

Bralca najprej seznanimo z uporabljenimi pristopi in tehnologijami. V nadaljevanju preučimo organizacijsko strukturo razvojne ekipe v podjetju, opišemo proces in postopek razvoja PO s poudarkom na testiranju ter tipični testni scenarij, po katerem je razvojna ekipa testirala programsko opremo. V naslednjem poglavju podamo oceno razvojnega procesa na podlagi modela stopenj zrelosti CMM. Sledi podrobnejša predstavitev orodja SoapUI in njegovih funkcionalnosti. Naslednje poglavje opiše testno okolje, testni scenarij, razvoj testnih primerov in izvedbo testiranja. Sledita poglavji, ki sta namenjeni priporočilom za izboljšavo procesa testiranja in sklepnim ugotovitvam.

Poglavje 2

Pregled področja

V poglavju so predstavljeni glavni pojmi, ki so potrebni za bralčevo razumevanje diplomske naloge.

2.1 Testiranje programske opreme

S pojavom prvih digitalnih računalnikov v začetku štiridesetih let in prvih programskih jezikov v začetku petdesetih let 20. stol. so se pojavili računalniški programi napisani v programskih jezikih, ki so se ukvarjali predvsem s problemi iz področja algoritmike in različnimi poslovnimi problemi. Z nastankom programskih jezikov pa pride tudi do napak in posledično do razhroščevanja programske kode (ang. code debugging). Razmejitev med razhroščevanjem in testiranjem postavi leta 1979 Glenford J. Myers [50] v svojih principih testiranja PO med katerimi je tudi "Testiranje je proces izvajanja programa z namenom odkritja napak."[53]. Testiranje lahko razdelimo na naslednje vrste [53]:

1. ***Testiranje enot oziroma testiranje modulov (ang. Unit testing)***
   Testiranje enot se osredotoča na posamezne dele programa. Ponavadi se prične s testiranjem najmanjših delov programa (modulov ali rutin). Kot primer lahko navedemo testiranje posameznih funkcij v program-

skem jeziku Java. Omenjen način nam olajša sprotno razhroščevanje, saj je samo iskanje napak omejeno le na funkcijo ali modul katerega testiramo in se tako obseg iskanja ob zaznavi napake znatno zmanjša.

2. ***Integracijsko testiranje (ang. integration testing)***
   Pri tej vrsti testiranja se manjše enote združijo in se testirajo kot celota. Cilj tovrstnega testiranja je odkriti napake v interakciji med posameznimi moduli oziroma enotami.

3. ***Funkcionalno testiranje (ang. functional testing)***
   Cilj funkcionalnega testiranja je odkriti morebitna odstopanja v delovanju programa od specifikacij. Specifikacije so opredeljene kot natančen opis delovanja programa iz vidika končnega uporabnika. V kontekstu funkcionalnega testiranja omenjamo tudi pristope s katerimi se največkrat izvaja. To so pristop bele škatle (ang. white box testing), pristop sive škatle (ang. grey box testing) in pristop črne škatle (ang. black box testing).

4. ***Sistemsko testiranje (ang. system testing)***
   Cilj sistemskega testiranja je odkrivanje napak v delovanju celotnega zaključenega integriranega sistema.

5. ***Sprejemno testiranje (ang. acceptance testing)***
   Sprejemno testiranje največkrat izvede naročnik, da preveri, če končni produkt ustreza specifikacijam.

6. ***Namestitveno testiranje (ang. installation testing)***
   Testiranje namestitve se izvaja, zatem ko je programska oprema že nameščena v ciljno okolje. Velikokrat lahko pride do napak med namestitvijo zaradi različnih razlogov, recimo: izbire različnih konfiguracij, nedosegljivosti zunanjih virov, odsotnosti omrežne povezave ali neprimerne strojne opreme. Cilj tovrstnega testiranja je zagotoviti, da programska oprema po namestitvi deluje pravilno in popolno.

2.2 Tehnologije in jeziki

2.2.1 XML označevalni jezik

Označevalni jezik XML (ang. Extensible Markup Language) je HTML-ju podoben samopisljivi označevalni meta-jezik[1] [44]. Razvit je bil z namenom, da preseže omejitve HTML jezika in je v ta namen zelo razširljiv [4]. XML poenostavlja prenos podatkov med sistemi, ki so sicer nekompatibilni.
XML lahko opredelimo tudi kot temelj, na katerem so zgrajene spletne storitve. Kot tak, opredeljuje definicijo podatkov in njihovo obdelavo [54]. Na podlagi formata XML je bilo razvitih več formatov dokumentov: RSS [19], SVG [25], XHTML [43] ter več industrijskih standardov (denimo HL7 [3] in OTA [13]).

2.2.2 SOAP protokol

Protokol SOAP (ang. Simple Object Access Protocol) je protokol namenjen strukturirani izmenjavi informacij med spletnimi storitvami v decentraliziranih distribuiranih okoljih [17]. SOAP protokol uporablja jezik XML in omogoča izmenjavo sporočil med različnimi operacijskimi sistemi.
Sestavljen je iz treh delov. Iz ovojnice (ang. envelope), množice pravil za šifriranje in konvencije o predstavitvi proceduralnih klicev in odzivov. SOAP ima tri glavne značilnosti: razširljivost, nevtralnost in neodvisnost.

2.2.3 REST arhitekturna načela

REST (ang. Representational State Transfer) je arhitekturni stil, ki definira množico omejitev za uporabo pri razvoju spletnih storitev [18]. Je brezstanjski, deluje pa na način odjemalec-strežnik. Uporablja standardne HTTP metode: GET, POST, PUT, DELETE in PATCH.

[1] Pri HTML jeziku so značke točno določene pri XML-ju pa si lahko imena značk poljubno izbiramo.

2.2.4 WSDL opisni jezik

Opisni jezik WSDL zagotavlja model in XML format za opis spletnih storitev [42]. WSDL lahko opredelimo tudi kot mehanizem za definicijo spletnih storitev, za oblikovanje SOAP zahtevkov z namenom komunikacije z določeno spletno storitvijo [54].
WSDL opisuje podatkovne tipe in strukturo spletne storitve. Ker ta opisni jezik temelji na jeziku XML je dovzeten za napade, zato je potrebna posebna skrb pri omogočanju dostopa do WSDL datotek [41].

2.2.5 Groovy skriptni programski jezik

Apache Groovy [34] je objektno usmerjen dinamični programski jezik, ki ga na svojem blogu prvič omeni James Strachan [45]. Lahko se ga uporablja tudi kot skriptni jezik za Java platformo, saj se dinamično prevaja v Java JVM strojni jezik in je tako kompatibilen z Java programsko kodo in Javanskimi knjižnicami. Večina javanske kode je sintaktično pravilna Groovy koda, kljub razliki v semantiki obeh jezikov. Enako velja za večino javanskih datotek, kar pomeni, da so tudi te veljavne Groovy datoteke.

2.3 Modeli in pristopi

2.3.1 Model stopenj zrelosti

Model stopenj zrelosti (ang. Capability Maturity Model) je bil razvit na Software Engineering Institute (SEI) po letu 1986 na zahtevo Ministrstva za obrambo ZDA [56]. Namen je bil razviti model, ki bi služil objektivnemu ocenjevanju procesa razvoja PO podizvajalcev Ministrstva za obrambo ZDA. Naslednik modela CMM je model CMMI, ki rešuje določene pomanjkljivosti modela CMM [55].

Model stopenj zrelosti predvideva pet stopenj zrelosti, ki predstavljajo sposobnost podjetja za razvoj kvalitetne programske opreme.

V vsaki stopnji pa se nahaja več procesnih področij, znotraj katerih se nadalje nahaja še veliko splošnih in posebnih ciljev.

1. stopnja - **Kaotičen proces**
   V organizaciji ne uporabljajo skoraj nobenega pristopa za obvladovanje razvojnega procesa PO znana sta le vhod in izhod. Proces ni dokumentiran.

2. stopnja - **Upravljan proces**
   Proces je dokumentiran, organizacija pa ima vzpostavljene osnovne procese vodenja projektov, ki omogočajo spremljanje stroškov, učinkovitost dela, in trajanje razvoja. Organizacija je zmožna ponoviti dosežene rezultate na podobnih projektih.

3. stopnja - **Definiran proces**
   Proces je potrjen kot standardni poslovni proces. Organizacija obvladuje tako upravljalske kot tudi tehnične aktivnosti. Te aktivnosti so dokumentirane in standardizirane v enoten proces razvoja PO v podjetju.

4. stopnja - **Kakovostno upravljan proces**
   Proces je kvantitativno upravljan v skladu z dogovorjenimi metrikami. Organizacija na tej stopnji spremlja svoj proces razvoja PO z natančnimi meritvami.

5. stopnja - **Optimizirajoč proces**
   Organizacija proces razvoja PO stalno izboljšuje in z uvajanjem novih idej in tehnologij ta proces stalno izboljšuje. Na tej stopnji je organizacija zmožna ovrednotiti vpliv sprememb.

2.3.2 Agilna metodologija SCRUM

Scrum je lahka agilna metodologija za upravljanje razvoja PO [20]. Scrum se kot procesno ogrodje uporablja že od zgodnjih devetdesetih. Metodologija sestoji iz: Scrum ekipa, definicija vlog v tej ekipi, dogodki, izdelki in

pravila. Pravila povezujejo vloge, dogodke in izdelke ter obvladujejo odnose med njimi.

Scrum ekipa je jedro metodologije. Ekipo sestavlja:

- **Lastnik produkta** (ang. product owner)
  Zadolžen za doseganje optimalne vrednosti produkta. Njegova naloga je, da razvojna ekipa razume naloge na prioritetnem seznamu (ang. backlog) in jih razporeja na prioritetni seznam teka, tako da kar najhitreje izpolnjuje zastavljene cilje.

- **Razvojna ekipa** (ang. development team)
  Ekipa se sestoji iz profesionalcev, ki so ciljno naravnani, visoko samoorganizirani in odgovorni. V očeh metodologije so vsi enakovredni. Odgovornost doseganja zastavljenih ciljev je na celotni ekipi in ne na posameznikih.

- **Skrbnik metodologije** (ang. scrum master)
  Skrbnik metodologije Scrum je odgovoren za promocijo metodologije v skupini [20]. To dosega, tako da pomaga ostalim razumeti vrednote in pravila metodologije v teoriji in praksi. Odgovoren pa je tudi za to, da ekipa spoštuje časovne omejitve posameznih SCRUM dogodkov.

Metodologija predvideva še dogodke, ki so:

- **Tek** (ang. sprint) Časovni okvir, ki je manjši od 30 dni. Med tekom nastane produkt, ki je pripravljen na uporabo in se ne uvaja sprememb, ki bi ogrozile zastavljene cilje SCRUM ekipe. Ob koncu enega teka se nemudoma prične nov tek.

- **Planiranje teka** (ang. sprint planning)
  Na dogodku sodelujejo vsi člani ekipe. Pomembni pa sta predvsem dve vprašanji: Kaj se lahko dokonča v planiranem teku in kako pridemo do tega. Planiranje teka je omejeno na maksimalno osem ur v posameznem teku.

- **Dnevni sestanki** (ang. daily SCRUM)
  Dnevni sestanki so omejeni na 15 minut in so na sporedu vsak dan teka. Namenjen je pregledu napredovanja ekipe oziroma oddaljenosti ekipe od zastavljenega cilja teka.

- **Pregled teka** (ang. sprint review)
  Pregled teka se izvede ob koncu vsakega teka. Izvaja se z namenom morebitnih prilagoditev prioritetnega seznama produktov. Na dogodku sodelujeta SCRUM ekipa in deležniki (ang. stakeholders), ki jih povabi lastnik produkta. Dogodek je časovno omejen na največ štiri ure na tek.

- **Presoja teka** (ang. sprint retrospective)
  Namen dogodka je identifikacija slabosti SCRUM ekipe in iskanje načinov za izboljšanje delovanja ekipe. Omejeno je na največ tri ure na tek.

V sklopu uporabe metodologije SCRUM nastanejo še naslednji izdelki:

- **Prioritetni seznam produktov** (ang. product backlog)
  Prioritetni seznam produktov je urejen seznam zahtev, ki jih ekipa mora implementirati v končni izdelek. Zanj je odgovoren lastnik produkta.

- **Prioritetni seznam teka** (ang. sprint backlog)
  Prioritetni seznam teka je množica nalog ali funkcionalnosti, ki jih je ekipa uvrstila v tek. Obsega delo, ki ga mora ekipa opraviti, da ob koncu teka doseže zastavljen cilj.

- **Dodana funkcionalnost** (ang. increment)
  Dodana funkcionalnost je vsota vseh dokončanih nalog ali implementiranih funkcionalnosti med trenutnim tekom in vsemi že opravljenimi teki. Dodana funkcionalnost je korak proti zastavljenemu končnemu cilju SCRUM ekipe.

2.3.3 Poker planiranje

Poker planiranje (ang. planning poker ali scrum poker) je tehnika agilnega razvoja PO, ki je namenjena skupinskemu ocenjevanju časa implementacije določene funkcionalnosti [48]. Pri izvajanju metode je prisotna celotna razvojna ekipa, vendar ocenjujejo samo razvijalci. Lastnik produkta (ang. product owner) pri ocenjevanju ne sodeluje.
Velikost ekipe naj bi bila do deset sodelujočih, sicer je bolje ekipo razdeliti na dve manjši ekipi. Za izvajanje tehnike potrebuje vsak sodelujoči posebne karte, ki so označene s številskimi vrednostmi. Velikokrat je to Fibonaccijevo zaporedje, ni pa pravilo. Ekipa se pred izvajanjem tehnike sporazume, če bo izločila oziroma prepovedala kakšno izmed vrednosti (denimo najmanjšo ali pa največjo). Proces ocenjevanja se začne, tako da moderator naglas prebere opis naloge in jo na hitro predstavi. Zelo pomembno je, da se vsi sodelujoči pri ocenjevanju zavedajo zahtevnosti implementacije. Po dogovorjenem času za razmislek vsak sodelujoči izbere karto in jo istočasno z drugimi razkrije. Ocenjevalca, ki sta podala najnižjo in najvišjo oceno predstavita razloge za svojo odločitev. Potem se ocenjevanje ponovi. Med tem procesom ekipa skupaj določi enotno oceno. Velikokrat se izkaže, da izkušena ekipa pride do tega po treh ciklih, kar pa ni pravilo.

2.3.4 Metoda KANBAN

Kanban temelji na ideji zaporednega izvajanja nalog. Število izvajanih nalog naj bo omejeno, tako da se novih nalog lotimo šele takrat, ko so trenutne dokončane [52].
Princip v proces razvoja PO vpelje omejitve števila nalog, ki jih razvojna ekipa upošteva v vsaki fazi procesa razvoja PO. Kanban se izkaže kot koristen tako za ekipe, ki razvijajo PO agilno, kot tudi za ekipe, ki razvijajo PO v skladu z bolj tradicionalnimi metodologijami razvoja PO.
Že najmanjša blokada na izvajani nalogi povzroči zamašek v sistemu. Če je teh blokad več, se delovni proces zaustavi in razvojno ekipo prisili, da problem

odpravi. Tipično razvojna ekipa za izvajanje metode Kanban uporablja tablo v fizični ali elektronski obliki. Tako lahko izpolnjevanje nalog spremljamo in merimo. Kanban lahko povzamemo v obliki naslednjih napotkov [52]:

- **Vizualiziraj tok dela**
  Razdeli večje dele na podnaloge in jih zapiši na lističe. Nalepi lsitiče na tablo. Uporabi stolpce, ki nosijo imena korakov posameznih faz v razvoju PO.

- **Omeji število nalog v delu**
  Vpelji število nalog, ki so v delu v posamezni fazi razvoja PO.

- **Meri čas dokončanja posameznih nalog**
  Meri čas, ki je potreben za dokončanje posameznih nalog in nato izračunaj povprečje.

2.3.5 Programiranje v parih

Programiranje v parih je tehnika agilnega razvoja PO, ki je del ekstremnega programiranja [46]. Pri programiranju sodelujeta dva razvijalca in uporabljata eno delovno postajo. Oseba, ki piše programsko kodo (ang. driver), se osredotoča predvsem na vnos programske kode, medtem ko mu opazovalec (ang. navigator) daje navodila in predloge na podlagi specifikacij ter je pozoren na morebitne napake v programski kodi. V večini primerov se pari sestavijo po stopnji izkušenosti. Sodelujoča občasno zamenjata vloge. Za tovrstno tehniko razvoja PO je bilo med drugim dokazano, da [47]: se poraba časa razvoja PO sicer zviša za 15%, vendar se s tem poveča kvaliteta programske kode, tehnično znanje, nivo medsebojne komunikacije sodelujočih in da prihaja do znatnega dviga zadovoljstva sodelujočih. Poleg tega se izboljša sposobnost reševanja zahtevnih problemov. Na tak način pa se nepretrgoma prenaša tudi znanje med sodelujočima.

Poglavje 3

Razvojna ekipa in proces testiranja

V poglavju je predstavljena struktura in organizacija razvojne ekipe E2[1], ki razvija in skrbi za IS eUprava.

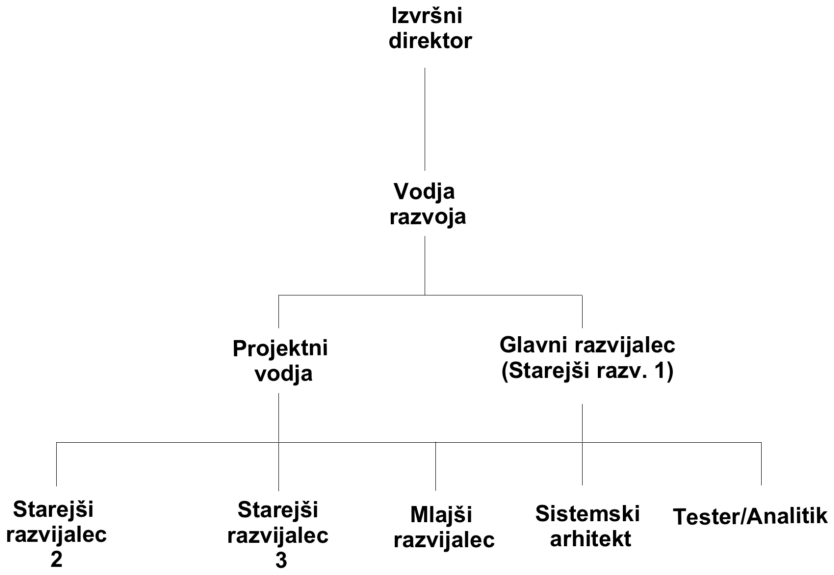
3.1 Organizacijska struktura razvojne ekipe

Ekipo sestavljajo:

- Projektni vodja, ki je zadolžen za komunikacijo in usklajevanje z naročnikom.
- Sistemski arhitekt, ki je zadolžen za načrtovanje.
- Izkušeni razvijalci (ang. senior developer) glavni razvijalec (izkušeni razvijalec 1), izkušeni razvijalec 2, izkušeni razvijalec 3. Njihova naloga je razvoj, pomagajo pa tudi sistemskemu arhitektu pri načrtovanju ko ta potrebuje pomoč. Glavni razvijalec je obenem tudi mentor izkušenemu razvijalcau2 in mlajšemu razvijalcu.
- Mlajši razvijalec (ang. junior developer), ki tudi sodeluje pri razvoju.

[1]Ime ekipe izvira iz trenutne verzije portala eUprava [14]. V času nastajanja diplomske naloge je bila verzija portala eUprava 2.0.

- Analitik/tester, ki izvaja ti dve spremljajoči dejavnosti [2]. Izkušeni razvijalci si po vlogah in izkušnjah niso enakovredni. Razvrščeni so po kriteriju staleža na projektu eUprava. Razvojno ekipo nadzira in upravlja vodja razvoja, ki je zadolžen za več podobnih ekip. Določeni člani ekipe sodelujejo tudi na ostalih projektih v podjetju.
  Hierarhija poročanja v organizacijski strukturi je vertikalna [39]: Člani razvojne skupine poročajo glavnemu razvijalcu in projektnemu vodji, ki oba poročata vodji razvoja (slika 3.1). Vodja razvoja poroča izvršnemu direktorju.



Slika 3.1: Komunikacijska hierarhija v ekipi E2

Skupina razvija programsko opremo v skladu z agilno metodologijo Scrum in uporablja sistem razporejanja nalog Kanban. Omenjeno kombinacijo metodologij formalno imenujemo Scrumban metodologija [38].

[2]Vloga analitika in testerja je v ekipi E2 združena.

3.2 Izvajanje SCRUM metodolgije v razvojni skupini

Metodologija SCRUM ne predpisuje uporabe specifičnih orodij. Posledično je izvajanje metodologije v različnih skupinah do neke mere specifično.

3.2.1 Vloge

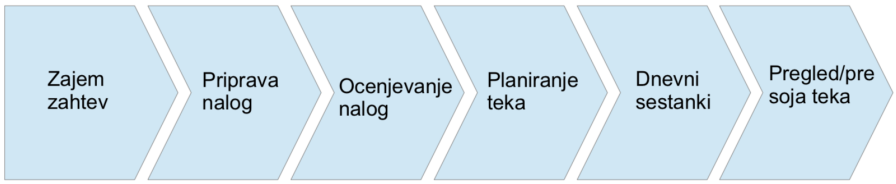
Projektni vodja je v vlogi skrbnika izdelka, sistemski arhitekt pa v vlogi skrbnika metodologije. Ostali člani skupine so v vlogi razvojne skupine. Skupina pozna tudi dodatno vlogov skrbnika KANBAN table, katero tudi zaseda tester/analitik.

3.2.2 Orodja, pripomočki in programska oprema

Razvojna ekipa uporablja pri izvajanju metodologije SCRUM orodje Atlassian Jira [30] z integrirano Kanban tablo. Uprablja pa se tudi fizična Kanban tabla z večbarvnimi samolepilnimi lističi in karte za poker planiranje z vrednostmi 0, 1/2, 1, 2, 3, 5, 8, 13, 20, 40, 100, ∞ . Kanban tabli nista povsem enaki. Obe sicer vsebujeta enake stolpce, ampak je na fizični tabli naveden naročnik dejansko delovna skupina sestavljena iz več članov. V digitalni verzije Kanban table ima vsak član naročnikove ekipe dodeljen svoj račun.

3.2.3 Razvojni proces PO v skupini

Razvojna ekipa izvaja Scrum metodologijo z fazami (slika 3.2):



Slika 3.2: Faze procesa razvoja PO v ekipi E2

1. ***Zajem zahtev (ang. Requirements gathering)***
   Zajem zahtev se izvaja v sodelovanju z naročnikom. Naročnik izrazi željo, da se implementira določena funkcionalnost. V začetnem koraku izvede projektni vodja sestanek z naročnikom, kjer mu naročnik predstavi vizijo in želje. V primeru obsežnejših funkcionalnosti je potrebno izvesti več sestankov, na katerih so prisotni tudi dodatni poslovni partnerji.

2. ***Priprava nalog na uvrstitev v tek (ang. Task grooming)***
   Projektni vodja izvede sestanek z analitikom ali (v primeru obsežne funkcionalnosti) s celotno razvojno skupino. Za raziskavo željene funkcionalnosti je zadolžen analitik, ki mora priskrbeti vse potrebne podatke. Nemalokrat predstavlja ta korak izziv analitiku, saj je za to nalogo običajno na voljo premalo časa.

3. ***Ocenjevanje pripravljenih nalog s pomočjo ocenjevalnega pokra (ang. planning poker)***
   Ocenjevanje poteka po pravilih ocenjevalnega pokra. Izločeni sta karti 0 in ∞. Na ocenjevanju je prisotna celotna skupina, kjer je projektni vodja v vlogi lastnika produkta. Za ocenjene naloge obstaja poseben seznam, oziroma fizična mapa, ki se uporabi v naslednjem koraku. V določenih primerih se uporablja tudi prilagajanje prioritetnega spiska (ang. backlog refinement).

4. ***Planiranje teka (ang. sprint planning)***

Trajanje teka je 21 dni. V tem koraku se iz posebnega seznama izberejo ocenjene naloge in uvrstijo na prioritetni seznam (ang. Sprint backlog). Ekipa se drži dogovora, da prioriteto predstavlja pozicija listka. Višja kot je pozicija, večja je prioriteta. Prioriteta nalog je ponavadi že vnaprej dogovorjena z naročnikom kot tudi datumi, ko mora posamezna funkcionalnost biti implementirana.

5. ***Dnevni sestanki (ang. daily scrum)***
   Dnevni sestanki (ang. daily scrum oziroma stand-up) potekajo vsak delovni dan ob 8.40. in se izvajajo stoje. Dnevni sestanek praviloma ne bi smel trajati več kot 15 minut [20] ampak se dogaja, da se včasih zavleče do 40 minut. Ker člani razvojne skupine delajo tudi na ostalih projektih v podjetju, včasih določeni član skupine dlje časa ne sodeluje v omenjeni skupini. Zato motijo in prekinjajo skrbnika metodologije in ostale člane z odvečnimi vprašanji.

6. ***Pregled in presoja teka (ang Sprint review in Sprint retrospective)***
   Pregled in presoja v okviru metodologije Scrum sicer nista enakovredna, ampak ju opisujemo kot en dogodek. Skupina obe obravnava na enem sestanku, kjer so prisotni vsi člani skupine. Začne se s pregledom teka. Nato vsak član na kratko predstavi najobsežnejšo ali najzanimivejšo opravljeno nalogo iz teka po svoji izbiri. Za tem je na vrsti presoja, ki se izvaja s pomočjo analogije jadrnice (ang. speedboat) [22]. Vsak član skupine našteje tri razloge sidra, ki po njegovem mnenju upočasnjujo razvoj (ang. anchors) in tri razloge vzpodbude, ki dajejo zagon (ang. gusts of wind).

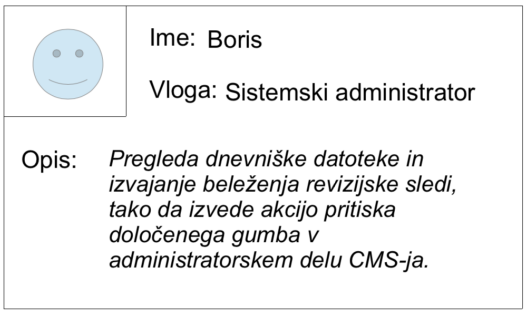
Kot dodatek, ki ga težko uvrstimo neposredno v Scrum metodologijo omenimo še, ti. škatlo zahval (ang. kudo box). Ta se odpre in izprazni ob vsakem končanem teku. Namen je vzpodbujanje kulture sodelovanja in dobrih medsebojnih odnosov v razvojni skupini. Člani vlagajo lističe s pohvalami sodelavca bodisi anonimno ali pa se vlagatelj podpiše. Pohvaljeni

sodelavec dobi na koncu teka simbolično nagrado.

3.3 Proces testiranja in tipični testni scenarij

Testiranje poteka po testnem scenariju, ki je bil razvit med prvo razvojno fazo portala eUprava 1.0 in se bistveno ne spreminja. Podpostopki testiranja (recimo konfiguracija okolja za testiranje JMS vrst) so zapisani v lokalni namestitvi baze znanja (ang. collaborative software). Postopek testiranja pred izdajo nove verzije tipično poteka v petih fazah (slika 3.4). V vsaki fazi se testirajo morebitne nove in že obstoječe (ang. legacy) funkcionalnosti:

1. **Ročno testiranje administratorskega dela portala.**
   V tej fazi tester izvede funkcionalno testiranje in uporabi vnaprej določene uporabniške vloge, ki so zapisane na papirnatih karticah [49]. Kartica vsebuje: ime, vlogo in akcije, katere izvede omenjena vloga (slika 3.3).



Slika 3.3: Primer kartice z opisom vloge in akcij

2. **Pregled že obstoječih naborov testov in njihova dopolnitev oziroma posodobitev.**
   Tester v tem koraku najprej izvede ukaz `svn update` in pridobi v svoj lokalni imenik najnovejšo verzijo SoapUI delovnega okolja. Do te faze je

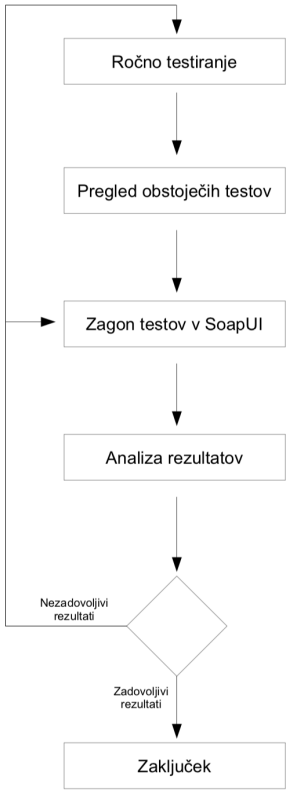
tester ustno že pridobil nekatere informacije in napotke s strani glavnega razvijalca [3]. V primeru, da se je uvajala nova funkcionalnost glavni razvijalec ustno preda navodila testerju in mu pomaga pri dopolnitvi testov. Kot že omenjeno sta vlogi analitika in testerja združeni, zato v primeru, ko ni novih funkcionalnosti, tester že sam ve, da ne potrebuje nobenih nadaljnih pojasnil ter lahko nadaljuje s tretjo fazo.

3. **Zagon testov v SoapUI.**
   Tester zažene nabore testov in spremlja potek izvajanja testov v programu SoapUI.
   V tej fazi tester opazuje izvajanje testov v skladu s testnimi scenariji. Če se le en test iz nabora testov ne izvede, se ne izvede celotni nabor testov.

4. **Analiza rezultatov testiranja in posvet z glavnim razvijalcem.**
   Tester predstavi rezultate testiranja in morebitne ovire glavnemu razvijalcu, ki mu nato nudi pomoč pri odpravi morebitne napake. Po potrebi glavni razvijalec sam dopolni teste. V primeru, da so popravki enostavni jih dopolni tester po navodilih glavnega razvijalca.

5. **Morebitna ponovitev postopka.**
   V primeru, da rezultati testiranja niso zadovoljivi, mora tester postopek ponavljati oziroma se vrača v fazi 1 ali 3 tako dolgo, dokler izkušeni razvijalec ustno ne potrdi, da je verzija pripravljena na predajo naročniku.

[3] Tester komunicira večinoma le z glavnim razvijalcem



Slika 3.4: Faze testiranja PO v ekipi E2

3.4 Dokumentiranje procesa

V opisanih fazah izdelava poročil o testiranju ni prisotna. V primeru morebitnih nejasnosti, se zato vprašanja naslavljajo na glavnega razvijalca ali arhitekta. Zaradi časovnih omejitev se velikokrat celotni proces testiranja lahko časovno precej podaljša.

Poglavje 4

Ovrednotenje procesa razvoja in testiranja PO na podlagi modela stopenj zrelosti CMM

V tem poglavju je opisana metodologija ocenjevanja in postopek ocenjevanja ter vprašalniki za prvo, drugo in tretjo stopnjo zrelosti po CMM. Vprašalniki vsebovani v uporabljeni metodologiji obsegajo skupno 101 vprašanj, ki so razdeljena v različne sklope. Na koncu je podana analiza rezultatov.

4.1 Metodologija

Za ovrednotenje procesa razvoja PO in testiranja smo izbrali *metodo za ovrednotenje zmožnosti razvoja PO za pogodbenike* (ang. Method for Assessing the Software Engineering Capability of Contractors) [51], ki je nastala na inštitutu SEI univerze Carnegie Mellon [27]. Sami smo postopek ovrednotenja malenkost prilagodili specifičnim okoliščinam in potrebam diplomske naloge. Ključna razlika je v tem, da smo ocenjevanje izvedli sami. To smo lahko storili, ker smo imeli neoviran dostop do ključnih kadrov v razvojni skupini in podjetju ter s tem tudi do vseh potrebnih informacij. S tem smo se izognili zahtevnemu sestavljanju interdisciplinarne skupine [51]. Odgovori

na vprašanja izhajajo iz opazovanja delovnega procesa in sodelovanja v razvojni skupini.
Sposobnost razvoja PO pogodbenika se s pomočjo vprašanj oceni preko treh ključnih področij:

- organizacijska struktura, viri, kadri in izobraževanje;
- proces razvoja PO in upravljanje tega;
- orodja in tehnologija.

Vprašanja temeljijo na naslednjih premisah: [51]:

- kvaliteta PO izhaja iz kvalitete procesa s katerim je bila ustvarjena,
- proces razvoja PO je lahko upravljan, merjen in se lahko vedno izboljša,
- na kvaliteto procesa razvoja PO vpliva tehnologija, ki ga uporablja, ter
- nivo tehnologije, mora biti primeren zrelosti procesa.

4.2 Postopek ovrednotenja zrelosti procesa razvoja programske opreme

Postopek ovrednotenja se izvede v treh korakih. Vsak naslednji korak se izvede, če prejšnji izpolni določene kriterije: [51]:

- V prvem koraku (stopnja 1) je bilo potrebno odgovoriti na splošna vprašanja iz področij, ki se osredotočajo na organizacijsko upravljanje in upravljanje z viri. Obsega sklope: organizacijska struktura, viri, kadri in izobraževanje ter tehnološko upravljanje. Podjetje se na začetku nahaja na prvi stopnji. Če odgovori pritrdilno na le nekatera izmed 17 vprašanj se še vedno nahaja na prvi stopnji.
- V drugem koraku (stopnja 2), mora biti delež pritrdilno odgovorjenih vseh vprašanj za stopnjo 2 najmanj 80%. Vprašalnik vsebuje tudi

posebna vprašanja (označena z zvezdico *). Delež pritrdilnih odgovorov na tovrstna vprašanja mora biti najmanj 90%. V primeru, da vse našteto velja, se organizacija nahaja na drugi stopnji in lahko nadaljujemo, sicer se organizacija nahaja na prvi stopnji.

- V tretjem koraku združimo vsoto pritrdilnih odgovorov na vprašanja za stopnji 2 in 3. Enako storimo z vprašanji, ki imajo zvezdico "*". Enako kot v drugem koraku, najmanj 80% delež pritrdilnih odgovorov na splošna vprašanja in najmanj 90% delež pritrdilnih odgovorov na posebna vprašanja. V tem primeru se organizacija nahaja na stopnji 3, sicer pa na stopnji 2. Za ocenjevanje stopenj 4 in 5 postopek ponovimo, kjer za ugotavljanje (denimo za stopnjo 5), združimo vse deleže pritrdilnih odgovorov stopenj 2, 3 in 4. Prag 80% in 90% se ne spremeni.

- V zadnjem koraku metodologija predvideva, da se oceni stopnja zrelosti, na kateri se nahaja organizacija kot celota. To pomeni, da je potrebno izračunati povprečje ocen več projektov, ki smo jih ocenjevali. Na tej točki smo v našem primeru metodologijo poenostavili in se osredotočili samo na ocenjevanje enega projekta in specifično na testiranje PO.

4.3 Vprašalnik 1

Osnovni vprašalnik je vseboval naslednja vprašanja [51]:

1. **Organizacijska struktura**

   (a) Ali ima vsak projekt iz področja razvoja PO, ki se izvaja v organizaciji, dodeljenega lastnega projektnega vodjo?

   (b) Ali projektni vodja razvojne skupine poroča direktno vodji razvoja?

(c) Ali zaposleni v razvojni ekipi, ki se ukvarja s področjem testiranja in kakovosti PO, poroča vodji razvoja po drugačni poti, kot razvijalci PO?

(d) Ali je v razvojni skupini določena vloga upravljalca kontrolnih vmesnikov?

(e) Ali so v skupini, ki se ukvarja z načrtovanjem sistemov, prisotni tudi inženirji razvoja PO?

(f) Ali za vsak projekt iz področja razvoja PO obstaja vloga skrbnika oziroma upravljalca sprememb PO?

(g) Ali v organizaciji obstaja ti. skupina za izboljšanje procesa razvoja PO oziroma SEPG (ang. Software Engineering Process Group) skupina?

2. **Viri, kadri in izobraževanje**

(a) Ima vsak razvijalec na voljo svojo delovno postajo?

(b) Ali v organizaciji obstaja obvezni uvajalni program oziroma šolanje za novo zaposlene razvojne menedžerje, da se seznanijo z upravljanjem razvoja PO?

(c) Ali v organizaciji obstaja obvezni uvajalni program oziroma šolanje za novo zaposlene razvijalce PO?

(d) Ali v organizaciji obstaja obvezni formalni program šolanja za novozaposlene menedžerje, ki bodo direktno nadzirali razvojne skupine, da se ti seznanijo s procesom razvoja PO?

(e) Ali v organizaciji obstaja obvezni formalni program šolanja za novozaposlene, ki bodo v vlogi vodij testiranja in kakovosti?

3. **Tehnološko upravljanje**

(a) Ali je v uporabi mehanizem za ozaveščanje zaposlenih v organizaciji o najnovejših tehnologijah za razvoj PO?

(b) Ali je v uporabi mehanizem, ki omogoča primerjavo in ovrednotenje tehnologij, ki jih uporablja organizacija v primerjavi s tistimi, ki so na voljo zunaj le-te?

(c) Ali je v uporabi mehanizem za odločanje o vpeljavi novih tehnologij v proces razvoja PO?

(d) Ali je v uporabi mehanizem za upravljanje in podporo vpeljave novih tehnologij?

(e) Ali je v uporabi mehanizem za odkrivanje in zamenjavo starih tehnologij?

4.4 Vprašalnik za drugo stopnjo

Iz vprašalnika za drugo stopnjo smo izbrali 7 vprašanj, ki se navezujejo samo na procesa razvoja in testiranja PO.

1. Ali se za vsak projekt izvajajo pregledi sleherne faze razvojnega proces PO?
2. Ali se v podjetju pri vsakem projektu razvoja PO uporabljajo standardi dobrega programiranja?
3. Je v uporabi mehanizem, ki poskrbi, da razvojna ekipa razume vsako nalogo, ki jo je potrebno implementirati?
4. Ali razvojna ekipa zbira statistične podatke o napakah med testiranjem PO?
5. Ali se vsaki najdeni napaki med testiranjem sledi od začetka do odprave?
6. Ali obstaja mehanizem sledenja spremembam v programski kodi (Kdo lahko izvaja spremembe programske kode in pod katerimi pogoji?)?
7. Obstaja mehanizem, ki poskrbi, da se redno izvajajo regresijska testiranja?

4.5 Vprašalnik za tretjo stopnjo

Iz vprašalnika za tertjo stopnjo smo izbrali 9 vprašanj, ki se navezujejo samo na procesa razvoja in testiranja PO.

1. Ali v podjetju obstaja obvezni formalni program za vodilne tehnične kadre na področju testiranja PO?
2. Ali podjetje uporablja standardiziran proces razvoja PO za vse projekte iz področja razvoja PO, ki je obenem tudi dokumentiran?
3. Ali vsebina imenikov, ki so vključeni v razvoj PO sledi kakšnim standardom?
4. Je v uporabi mehanizem za ocenjevanje zmožnosti ponovne uporabe že napisane programske kode?
5. Ali priprava testov enot sledi kakšnim standardom?
6. Ali ekipa beleži statistiko napak v implementaciji programske kode?
7. Ali je v uporabi kakšen mehanizem, ki poskrbi za skladnost s standardi razvoja PO?
8. Je v uporabi mehanizem, ki skrbi za kvaliteto regresijskega testiranja?
9. Se izvajajo formalni pregledi testnih primerov?

4.6 Rezultati

V tabeli na sliki 4.1) so navedeni odgovori, na osnovna vprašanja, ki preverjajo prvi zrelostni nivo.

Vprašanje	Odgovor
1.a	NE
1.b	DA
1.c	NE
1.d	DA
1.e	NE
1.f	NE
1.g	NE
2.a	DA
2.b	NE
2.c	NE
2.d	NE
2.e	NE
3.a	NE
3.b	NE
3.c	NE
3.d	NE
3.e	NE

Slika 4.1: Odgovori na vprašalnik 1

Izmed odgovorov na 17 osnovnih organizacijskih vprašanj smo zabeležili le 3 pritrdilne (17.64%), ostalih 14 pa je bilo nikalnih. Pogoj za napredovanje v sklop vprašanj za zrelostni nivo 2 tako ni izpolnjen. Čeprav smo predstavili tudi oba sklopa vprašanj za doseganje druge in tretje stopnje na njih nismo odgovarjali, saj podjetje ne izpolnjuje niti osnovnih kriterijev. Ocenjevanje le enega projekta tega rezultata ne spremeni; tudi če bi jih ocenjevali več, bi bili tovrstni odgovori enaki. Ker smo ocenjevali le en projekt, nam ni potrebno izračunati povprečja, kar pomeni da se organizacija nahaja na stopnji 1 modela stopenj zrelosti CMM.

Za to stopnjo je značilno, da podjetje ne uporablja nobenega pristopa za obvladovanje razvojnega procesa. Znan je le vhod in izhod, proces razvoja PO pa je zato naključen [56].

Večina odgovornosti leži na glavnem razvijalcu (starejši razvijalec 1), ki je odgovoren za vse vidike procesa razvoja PO. Glavni razvijalec poseduje največ tehničnega znanja o portalu eUprava in je tako nepogrešljiv člen razvojne ekipe na katerem sloni odgovornost vsake faze razvoja PO.

Tovrstna odvisnost od le enega razvijalca pa je tudi očitna slabost. V primeru daljše odsotnosti to lahko pomeni katastrofalne posledice za projekt [2].

Poglavje 5

Programsko orodje SoapUI

V tem poglavju je opisano odprtokodno orodje za testiranje PO SoapUI [33] s pomočjo katerega se v razvojni ekipi E2 izvajajo osrednja testiranja PO, natančneje testiranja IS eUuprava. Namen te predstavitve je osvetliti praktični vidik izvajanja testiranja v podjetju.
Poglavje opisuje predvsem glavne funkcionalnosti, ki jih ogrodje omogoča. Večina vsebine tega poglavja je povzeta po uradni dokumentaciji orodja SoapUI [26, 36].

5.1 Opis orodja SoapUI

Orodje SoapUI spada med odprtokodna orodja, ki so izdana pod licenco EUPL [1]. Namenjeno je testiranju aplikacij, ki slonijo na storitveno usmerjeni arhitekturi (ang. SOA) in arhitekturi za izmenjavo podatkov med spletnimi storitvami (ang. REST). Razvito je v programskem jeziku Java [35], uporabniški vmesnik pa je zgrajen s pomočjo knjižnice Swing [15]. Prva verzija je bila izdana leta 2005, trenutna verzija je 5.4. Uporabljajo ga podjetja Apple, Vodafone, SalesForce in Microsoft [33]. Uporabljena verzija je brezplačna, obstaja pa tudi plačljiva verzija SoapUI PRO (ReadyAPI), ki vsebuje še dodatne funkcionalnosti.
SoapUI omogoča več vrst testiranja: funkcionalno, obremenitveno, varno-

stno, podatkovno usmerjeno testiranje (ang. data driven testing). Omogoča tudi funkcionalnosti kot izdelavo nastavkov SOAP in REST storitev (ang. mocking), napredno avtomatizacijo testnih scenarijev s programskim jezikom JavaScript ali Groovy ter nadzor HTTP sej.

5.2 Delovno okolje in projekti v orodju SoapUI

SoapUI ponuja dva formata projektov: samostojni (ang. standalone) in sestavljeni (ang. composite) projekt. Prvi je standardni format, ki je predviden za uporabo v enouporabniškem načinu, drugi pa omogoča sodelovanje več uporabnikom na istem projektu. Projekti se nahajajo v delovnem okolju (ang. workspace). Okolja lahko uporabnik prilagaja različnim testnim nastavitvam, strankam in sistemom.
Orodje podpira tri tipe projektov:

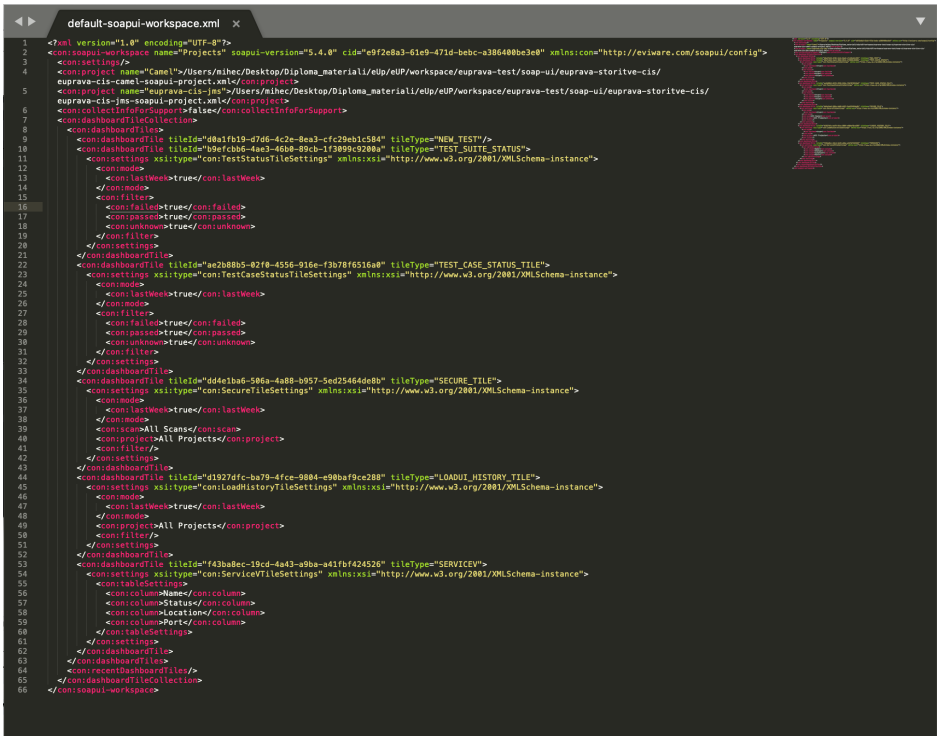
- `SOAP projekt`
  Ustvarjen na podlagi WADL datoteke ali direktno s pomočjo spletnega naslova in parametrov le-tega. Testira se lahko vsako področje SOAP storitve in tudi uporabo standardov WS-Security [12], WS-Addressing [40] in MTOM [21].

- `REST projekt`
  Ustvarjen na podlagi WADL datoteke ali direktno preko spletnega naslova in parametrov le-tega.

- ***Generični projekt***
  Generični projekt običajno vsebuje tako nabore testov za testiranje REST storitev kot tudi nabore testov ali posamezne teste za testiranje SOAP storitve.

Delovno okolje je shranjeno v obliki XML datoteke, v kateri se nahajajo projekti in vsi izdelki projektov. Izdelki so: vmesniki, testi, zbirke testov, testne

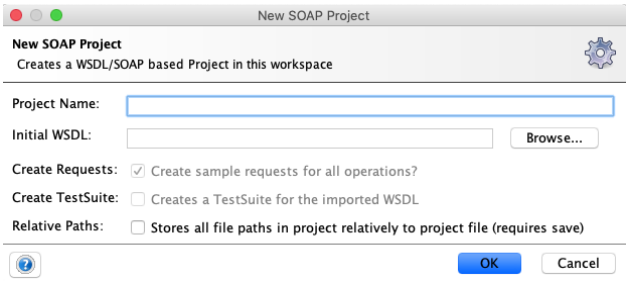
skripte inpd. Na sliki 5.1 vidimo projekt v XML obliki.



Slika 5.1: Delovno okolje shranjeno v formatu XML

5.2.1 Kreiranje SOAP projekta

Nov projekt tipa SOAP lahko uporabnik ustvari, tako da v meniju `File` izbere možnost `New SOAP Project` ali pa s klikom na bližnjico (ikona imenika z napisom SOAP in zvezdico) v orodni vrstici bližnjic.

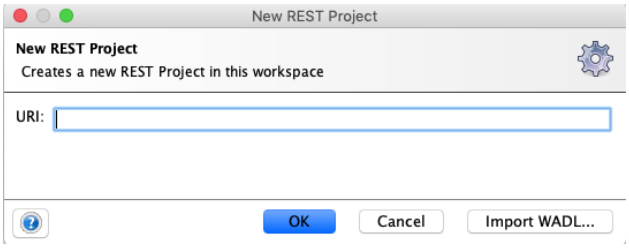


Slika 5.2: Okno za ustvarjanje novega SOAP projekta

Pri tem izbere tudi ime projekta, WSDL datoteko (oziroma pot do datotek) in se odloči ali naj se avtomatično ustvarijo zahtevki za vse definirane operacije v WSDL datoteki (slika 5.2).

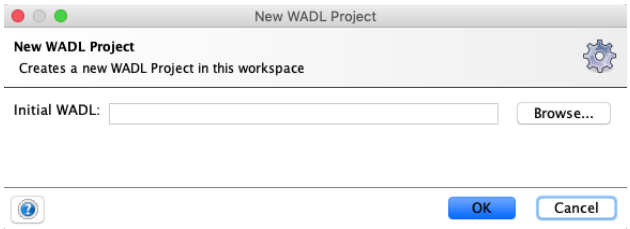
5.2.2 Kreiranje REST projekta

Nov projekt tipa REST lahko uporabnik ustvari, tako da v meniju `File` izbere možnost `New REST Project` ali pa s klikom na bližnjico (ikona imenika z napisom REST in zvezdico) v orodni vrstici bližnjic.



Slika 5.3: Ustvarjanje novega projekta iz spletnega naslova

Program uporabniku ponudi možnost ustvarjanja projekta s pomočjo spletnega naslova storitve (slika 5.3). Obstaja tudi možnost ustvarjanja projekta s pomočjo uvoza WADL datoteke (slika 5.4).



Slika 5.4: Uvoz WADL datoteke

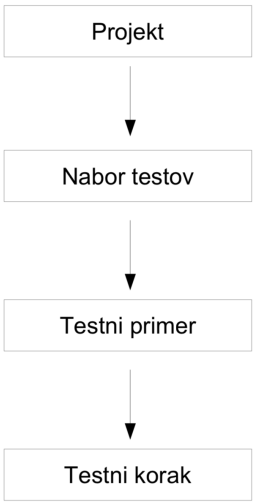
5.2.3 Kreiranje sestavljenega projekta

Nov sestavljen projekt lahko uporabnik ustvari, tako da v meniju `File` izbere možnost `New Empty Project` ali s klikom na bližnjico (ikona praznega imenika z zvezdico) v orodni vrstici bližnjic.

5.3 Hierarhija glavnih gradnikov

Glavne gradnike orodja SoapUI predstavljajo:

- projekt (ang. project),
- nabor testov (ang. test suite),
- testni primer (ang. test case),
- testni korak (ang. test step).



Slika 5.5: Hierarhija glavnih gradnikov orodja SoapUI

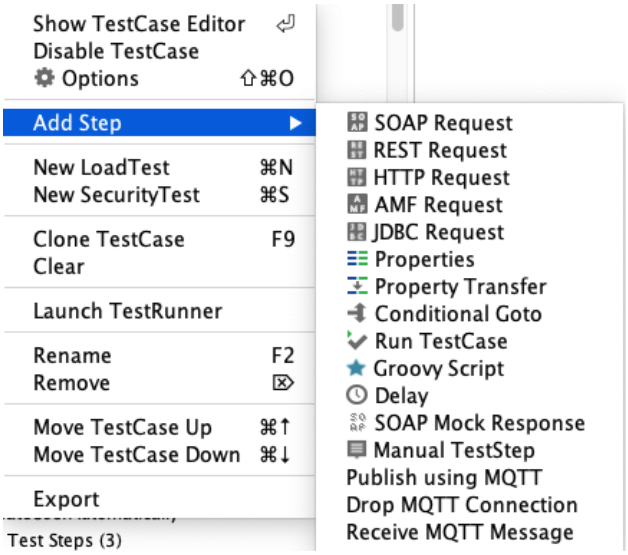
Gradniki so razporejeni po hierarhiji (slika 5.5). Testni koraki so vsebovani v testnem primeru, ki je vsebovan v naboru testov, ti pa so vsebovani v projektu.

5.3.1 Tipi testnih korakov

Testni korak (ang. test step) je osnovni gradnik in kot tak tudi najbolj pogosto uporabljen. Gradnike delimo na pet glavnih kategorij:

- validacija funkcionalnosti storitev, ki obsega:
  - SOAP zahtevek,
  - REST zahtevek,
  - JDBC zahtevek,
  - AMF zahtevek,
  - HTTP zahtevek ali
  - odziv imitiranega servisa (ang. mock response).

- testni koraki vezani na lastnosti projekta/nabora testov, podrobneje:
  - lastnosti (ang. properties),
  - prenos polja lastnosti (ang. property transfer).
- testni koraki vezani na dostop do podatkov, torej:
  - podatkovni vir (ang. data source),
  - zanka podatkovnega vira (ang. data source loop),
  - vpis podatkov v zunanji vir (ang. data sink).
- testni koraki vezani na tok izvajanja:
  - pogojni skok (ang. conditional GOTO),
  - zamik (ang. delay),
  - zagon testnega primera (ang. run testcase),
  - zanka podatkovnega vira (ang. data source loop).
- razni testni koraki (ang. miscellaneous), ki so predstavljeni kot skripta, oziroma skriptni testni korak.

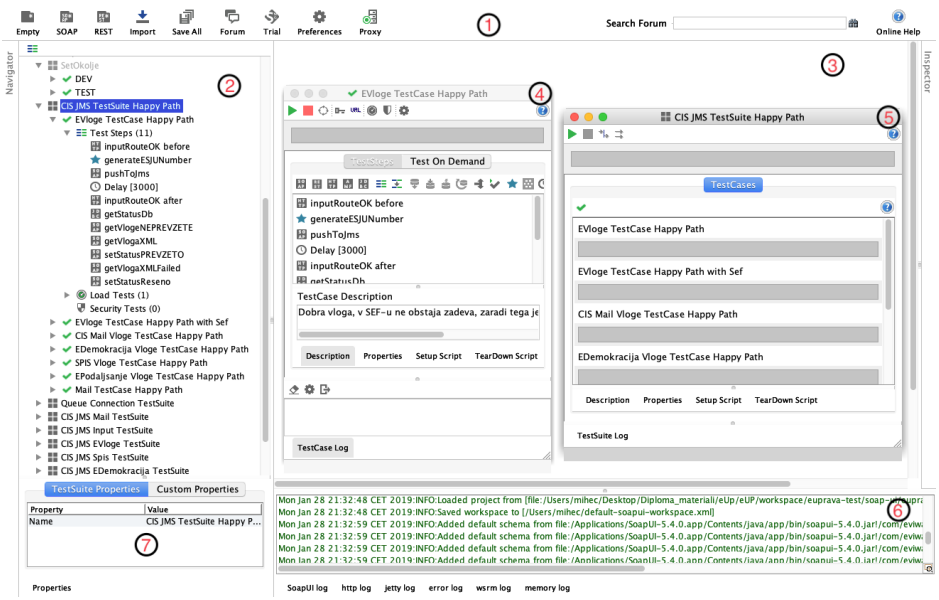


Slika 5.6: Menu izbire testnega koraka

Na sliki 5.6 vidimo postopek izbire in dodajanja testnega koraka.

5.4 Uporabniški vmesnik

Uporabniški vmesnik (slika 5.7) sestavlja:



Slika 5.7: Uporabniški vmesnik orodja SoapUI

1. ***Orodna vrstica bližnjic (ang. icon shortcuts bar)***
   Ikone v orodni vrstici so bližnjice do akcij, ki jih lahko izvede uporabnik.

2. ***Vodič (ang. navigator pane)***
   Vodič je glavni menu vseh odprtih in zaprtih projektov, na katerih dela uporabnik. Posamezni projekt je predstavljen kot mapa, v kateri so hierarhično urejeni vmesniki, nabori testov, testi in testni koraki.

3. ***Delovna površina***
   Na delovni površini uporabnik tvori nabore testov, teste in testne korake. Vsa ostala okna se odpirajo na tej površini, zato je tudi glavna delovna površina orodja.

4. ***Okno testa***
   V tem oknu lahko uporabnik sestavi test iz več testnih korakov in ga poganja ali ustavlja. Na voljo ima različne gradnike: ustvarjanje novega SOAP zahtevka, ustvarjanje novega REST zahtevka, ustvarjanje novega AMF zahtevka, ustvarjanje novega JDBC zahtevka in tudi nekaj

ostalih gradnikov, ki so vezani na prenose vrednosti med posameznimi testnimi koraki.

V spodnjem delu okna se nahaja podokno, v katerem je mogoče ustvariti opis testa. Možno je nastaviti še ti. vzpostavitveno skripto (ang. setup script) in skripto, ki se izvede po zadnjem testnem koraku (ang. teardown script). Iz obstoječega testa se lahko ustvari tudi obremenitveni test, pri katerem lahko nato nadalje določa število niti (ang. threads) in različne strategije ter zamik med posameznimi zahtevki. Okno pa vsebuje še podokno imeniškega zapisa, v katerem uporabnik v živo spremlja potek izvedbo testnih korakov.

5. ***Okno nabora testov***

   Okno nabora testov vsebuje posamezne teste. Uporabnik ima možnost poganjati ali ustavljati teste in določiti, njihovo zaporedno ali vzporedno izvajanje. Tudi tukaj ima možnost spreminjati hierarhijo testov.

   Enako kot v oknu posameznega testa ima uporabnik možnost ustvariti skripto, ki se zažene po zadnjem testu. Okno vsebuje še podokno imeniškega zapisa, v katerem uporabnik spremlja potek izvajanja testov.

6. ***Pregledovalnik dnevniških zapisov (ang. log inspectors)***

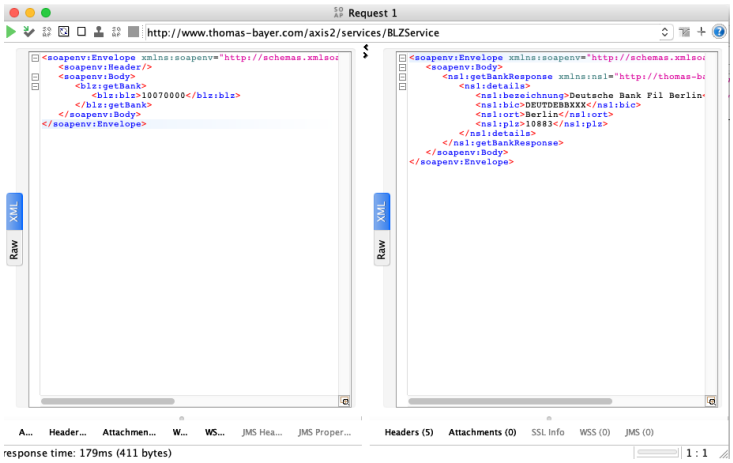
   Pregledovalnik dnevniških zapisov omogoča uporabniku spremljanje dogodkov iz kategorij: SoapUI dogodki (ang. SoapUI log), http dogodki (ang. http log), jetty dogodki (ang. jetty log), splošne napake (ang. error log), wsrm dogodki (ang. wsrm log), poraba pomnilnika (ang. memory log).

7. ***Okno nastavitev projekta/delovnega okolja***

   Okno nastavitev se nadalje deli na privzete nastavitve in na nastavitve po meri (ang. custom properties). Te omogočajo uporabniku, da spreminja privzete nastavitve ali dodaja svoja polja in spremenljivke, ki so lahko globalne ali pa lokalne. Tipično se v teh poljih določajo spremenljivke (podatki za avtentikacijo, enolični določnik seje).

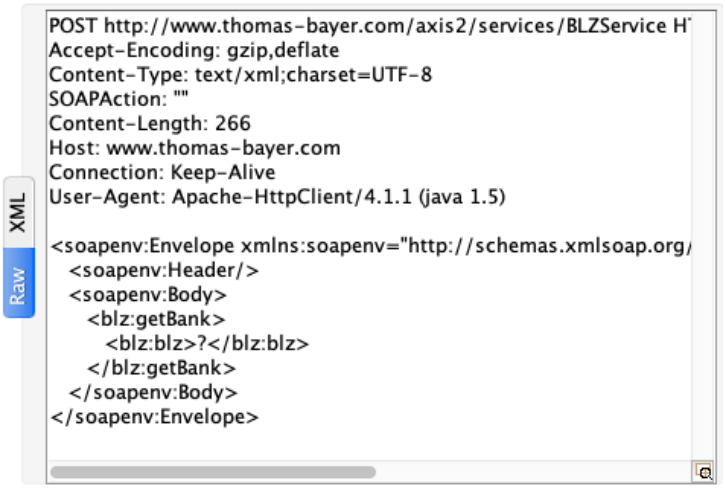
5.4.1 Glavno okno SOAP zahtevka

Glavno okno SOAP zahtevka je sestavljeno iz treh podoken, izmed katerih je eno opcijsko. Levo podokno je namenjeno oblikovanju klica v XML obliki, podpira pa tudi oblikovanje v surovi obliki (ang. raw), kjer je prisotno tudi zaglavje zahtevka (sliki 5.8 in 5.9). Iz zahtevka je mogoče neposredno ustvariti testni korak in ga dodati k izbranemu testnemu primeru.



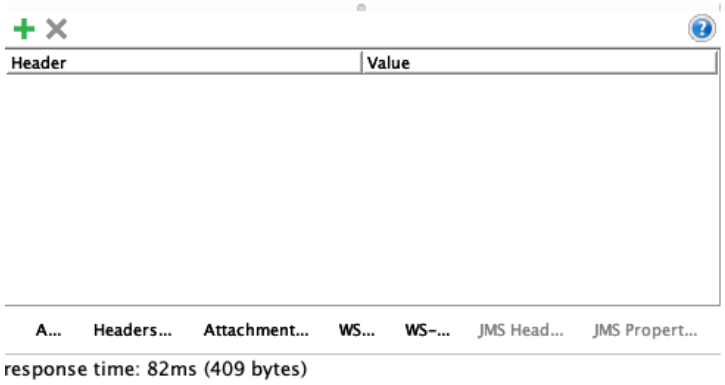
Slika 5.8: Glavno okno SOAP zahtevka

Desno podokno je enako levemu, vendar je namenjeno odgovorom na zahtevke.



Slika 5.9: SOAP zahtevek v surovi obliki

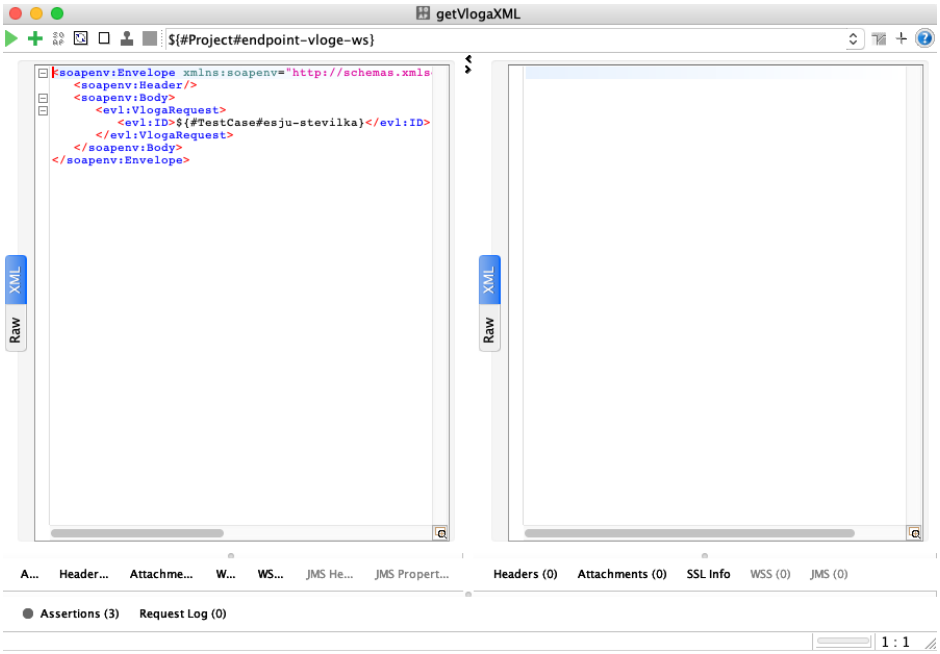
Opcijsko okno se lahko odpre pod levim podoknom, če želi uporabnik določiti oziroma spreminjati nastavitve: avtentikacije, zaglavja, WS-A naslavljanja, WS-Reliable sporočanja ali JMS zaglavja (slika 5.10).



Slika 5.10: Opcijsko podokno

5.4.2 Okno testnega koraka SOAP

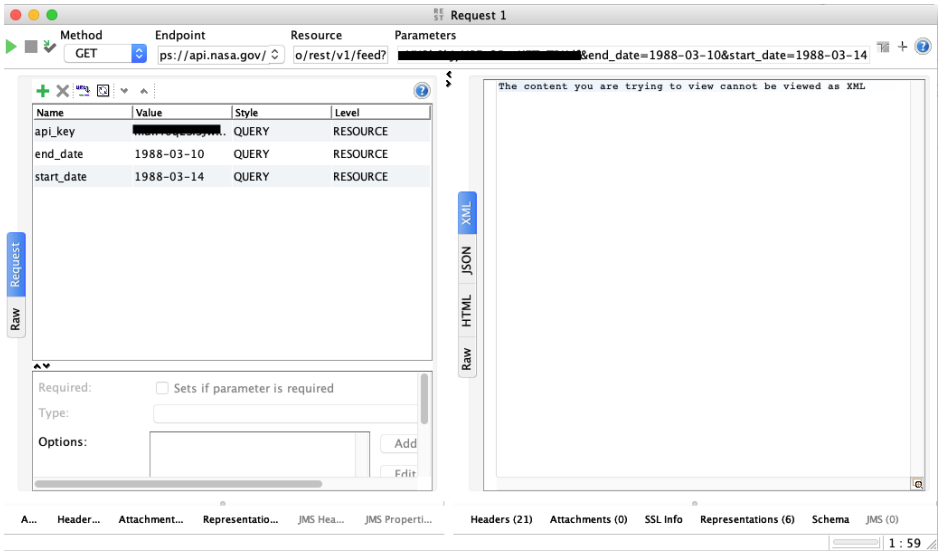
Okno testnega koraka je zelo podobno glavnemu oknu za ustvarjanje in konfiguracijo SOAP zahtevka. Razlikuje se v tem, da omogoča uporabniku dodajanje validacijskih točk (ang. assertion) (slika 5.11).



Slika 5.11: Testni korak SOAP zahtevka

5.4.3 Glavno okno REST zahtevka

V tem oknu lahko uporabnik izvaja različne klice na REST storitve in nastavlja različne parametre za dostop do programskega vmesnika.
Okno je sestavljeno iz treh podoken (izmed katerih sta dve opcijski), izbirnika HTTP metod, izbirnika končne točke, orodne vrstice za nastavitev dodatnega vira in orodne vrstice, združenih parametrov (slika 5.12).
Levo okno omogoča pogled v surovi obliki ali pa pogled zahteve. V tem pogledu lahko uporabnik dodaja pare: ime parametra in vrednost. Pod oknom se nahaja desno okno. To je namenjeno odgovorom na zahtevke. Ti pa so lahko v različnih oblikah: surova, HTML, JSON ali pa XML. Pod oknom se nahajajo nastavitve za zaglavje, priponke, SSL sejo, JMS zaglavje.



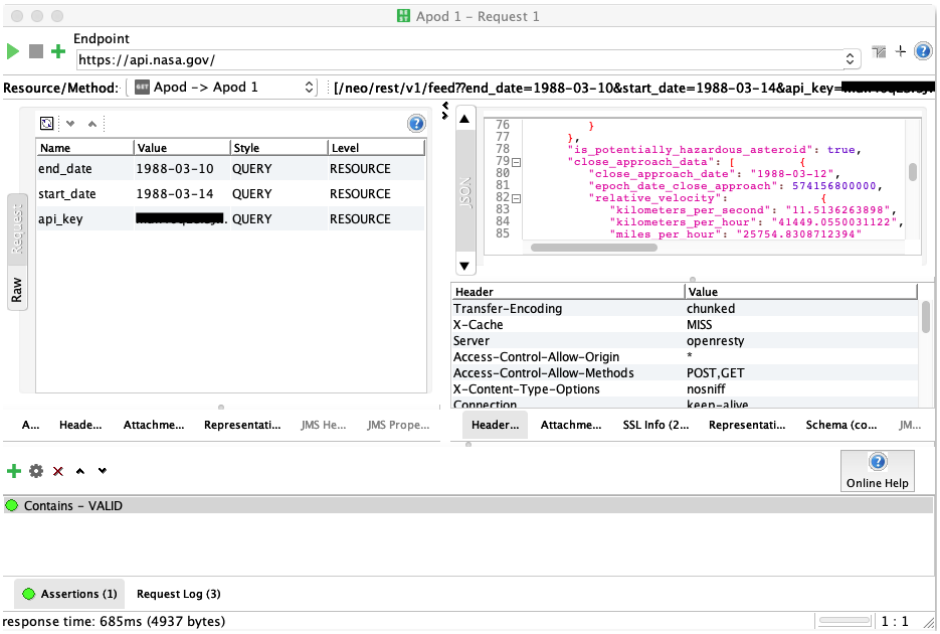
Slika 5.12: Glavno okno REST zahtevka

Okno se odpre pod levim podoknom, če želi uporabnik določiti ali spreminjati nastavitve avtentikacije, zaglavja ali priponk.

Kot pri glavnem SOAP oknu je tudi v glavnem REST oknu prisotna možnost neposrednega ustvarjanja testnega koraka in dodajanja k testnemu primeru.

5.4.4 Okno testnega koraka REST

Okno testnega koraka REST lahko razdelimo na pet podoken (slika 5.13). Dve sta opcijski, tri pa stalno prisotna. Na voljo je tudi izbirnik metode in vrstica, kjer so definirani parametri, ki se pošljejo ob zahtevku. Na levi strani se nahaja podokno za parametre v obliki ime parametra in vrednost. Mogoč je tudi pogled v surovi obliki. Desno podokno je namenjeno prikazu odgovora REST storitve. Podpira štiri različne prikaze: surovega, HTML, JSON in XML. Pod omenjenim oknom se nahaja še opcijsko okno, kjer je mogoče spreminjati nastavitve: zaglavja, priponk in SSL povezave.



Slika 5.13: Testni korak REST zahtevka

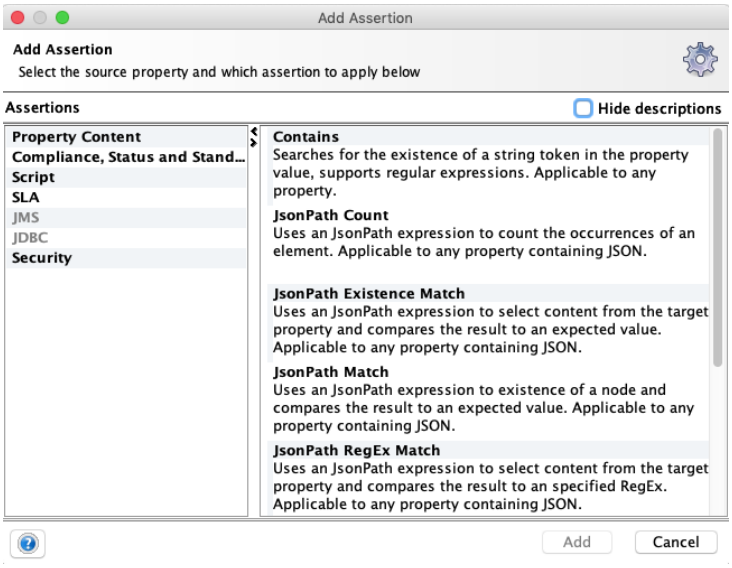
V spodnjem oknu je možno dodajati validacijske točke in spreminjati vrstni red le-teh. Tudi tukaj je možno ustvariti testni korak in ga dodati k testnemu primeru.

5.5 Podprti tipi validacijskih točk

V orodju SoapUI večina operacij vključuje analizo odziva spletne storitve v testiranju. Gre za odziv v XML, JSON, surovi ali pa HTML obliki. Za preverjanje, če se v odgovoru nahajajo pričakovani podatki, poznamo v orodju SoapUI koncept validacijskih točk (ang. assertion). Te točke nam omogočajo potrjevanje podatkov, ki jih pričakujemo v odgovorih spletnih storitev.

5.5.1 Kategorije validacijskih točk

Orodje SoapUi pozna več vrst validacijskih točk, ki so zaradi prijaznosti uporabniku razdeljene v več kategorij (slika 5.14).



Slika 5.14: Okno za izbiro validacijske točke

Vsebinske validacijske točke

Kategorija vsebinskih validacijskih točk (ang. property content) je namenjena preiskovanju vsebine prejetega odgovora v testnem koraku. Vsebuje kriterije: *vsebuje (ang. contains)*, *ne vsebuje (ang. not contains)*, *XPath ujemanje (ang. XPath match)*, ter *XQuery ujemanje (ang. XQuery match)*.

Statusi skladnosti in standardi

- **Prenos vseh virov**
  Prenos vseh virov HTML dokumenta in potrditev, da so vsi dosegljivi.
- **Neveljavni HTTP statusi**
  Preveri ali je testni korak prejel HTTP statusno kodo, ki je na seznamu dovoljenih statusnih kod.
- **Izključitev krivde SOAP odziva**
  Potrditev, da zadnje prejeto SOAP sporočilo ne vsebuje napake.

- **Skladnost s shemo**
  Potrdite, da je zadnje prejeto sporočilu v skladu z WSDL ali WADL shemo.

- **SOAP krivda**
  Potrditev, da zadnje prejeto SOAP sporočilo vsebuje napako.

- **SOAP odziv**
  Potrditev, da je zadnji prejeti odziv tipa SOAP veljaven.

- **Veljavni HTTP statusi**
  Preveri ali je testni korak prejel HTTP statusno kodo, ki je na seznamu dovoljenih statusnih kod.

- **Zahtevek WS naslavljanja**
  Potrdi, da zadnji prejeti WS zahtevek vsebuje veljavno zaglavje.

- **Odziv WS naslavljanja**
  Potrdi, da zadnji prejeti WS odziv vsebuje veljavno zaglavje.

- **Status WS varnosti**
  Potrdi, da zadnje prejeto WS sporočilo vsebuje veljavno zaglavje.

Skripta

Vsebuje le eno kategorijo, ki omogoča uporabniku, da opravi poljubne validacije s pomočjo skripte.

SLA

SLA kategorija vsebuje le en kriterij. Odziv SLA (ang. Response SLA), ki preveri ali je bil zadnji prejeti odziv v časovnem okviru, ki je bil zastavljen s strani uporabnika.

JMS

JMS kategorija vsebuje dva kriterija: JMS status, ki preveri ali se je testni korak JMS uspešno izvedel in JMS premor (ang. JMS timeout), ki preveri ali se JMS zahtevek ni izvajal več časa, kot je uporabnik določil.

JDBC

Tudi JDBC kategorija vsebuje dva kriterija: JDBC status, ki preveri ali se je testni korak uspešno izvedel in JDBC premor (ang. JDBC timeout), ki preveri ali se JMS zahtevek ni izvajal več časa, kot je uporabnik določil.

Varnostne validacijske točke

Kategorija vsebuje kriterij - razkritje občutljivih informacij (ang. sensitive information disclosure), ki ujame morebitno razkritje občutljivih podatkov testiranega sistema.

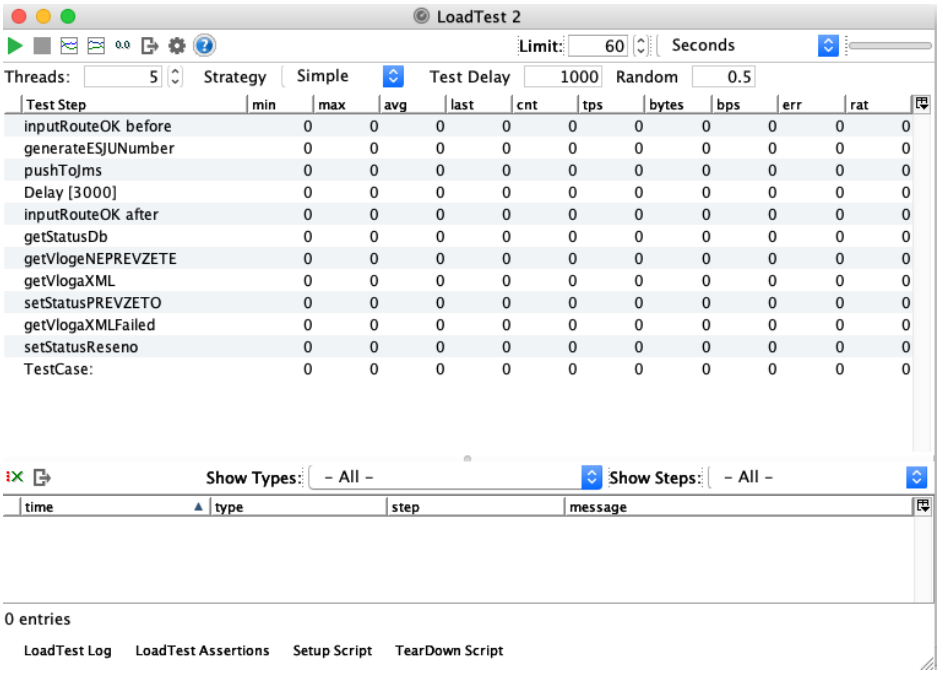
5.6 Glavne funkcionalnosti

Funkcionalno testiranje

Funkcionalno testiranje je osrednji namen orodja SoapUI. Orodje s pomočjo množice različnih gradnikov (REST zahtevek, SOAP zahtevek, prenos lastnosti, skriptiranje...) omogoča uporabniku razvoj kompleksnih testnih primerov in združevanje v nabore testov.

Performančno testiranje

Obstaja več vrst performančnih testiranj. Na tem mestu bomo na kratko predstavili le funkcionalnost izvajanja obremenitvenega testiranja, kjer uporabnik preverja obnašanje sistema pod visoko obremenitvijo (slika 5.15). Medtem lahko meri odzivne čase, propustnost ali pa opazuje različne parametre sistema. Obremenitvene teste je mogoče ustvariti neposredno iz okna nabora testnih primerov.



Slika 5.15: Uporabniški vmesnik za izvajanje obremenitvnenih testov

Vmesnik za izvajanje obremenitvenega testiranja omogoča spreminjanje nastavitev števila niti, strategije testiranja, časovnega zamika med posameznimi klici ter skupni čas izvajanja testiranja. Nastavitve omejevanja lahko spremenimo, izbiramo lahko med časom trajanja v sekundah, skupnim številom ponovitev testiranj in število testiranj na posamezno nit.
Na desni zgornji strani se nahaja tudi vrstica napredka, spodaj pa okno z dnevnikom izvajanja testiranj (ang. load test log) in validacijske točke. Dodati pa je mogoče tudi skripti, ki se izvedeta pred in/ali po testiranju.
Rezultate je v omejeni obliki možno izvoziti v XML obliko.

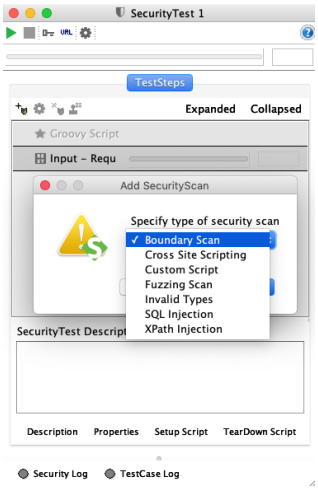
Varnostno testiranje

Orodje omogoča uporabniku ustvarjanje in izvajanje množice različnih varnostnih testov (oziroma naborov varnostnih testov) s katerimi lahko uporabnik preveri varnost testirane aplikacije. Varnostni testi se dodajajo k posameznemu testnemu koraku (slika 5.16). Orodje podpira več kategorij varnostnih

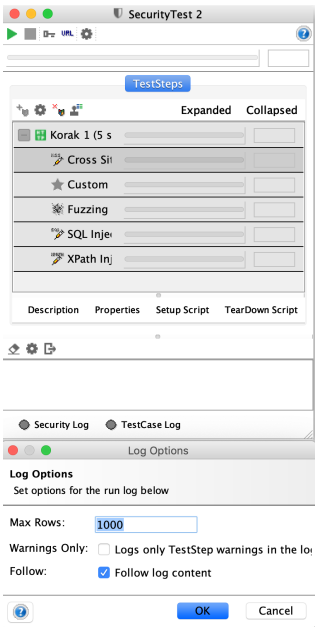
testiranj:

- vrivanje SQL stavkov (ang. SQL injection),
- napad na spletno stran ali posamezna polja z vrivanjem zlobne kode v skriptnem jeziku (ang. XSS),
- napad oziroma vrivanje s pomočjo poizvedovalnega jezika XPath (ang. XPath injection),
- vrivanje neveljavnih, nepričakovanih ali naključnih podatkov kot vhod v program (ang. fuzzing),
- XML bomba (ang. XML bomb),
- zlonamerna priponka (ang. malicious attachment),
- nepravilno oblikovan XML (ang. malformed XML),
- neveljavni tipi (ang. invalid types),
- nepričakovani vhodni podatki (ang. boundary scan).

Vseh kategorij ni moč dodati k zahtevkom vsakega tipa. Varnostna testiranja so možna le za testne korake tipa SOAP in HTTP. Dodamo lahko več različnih varnostnih testov (slika 5.17).



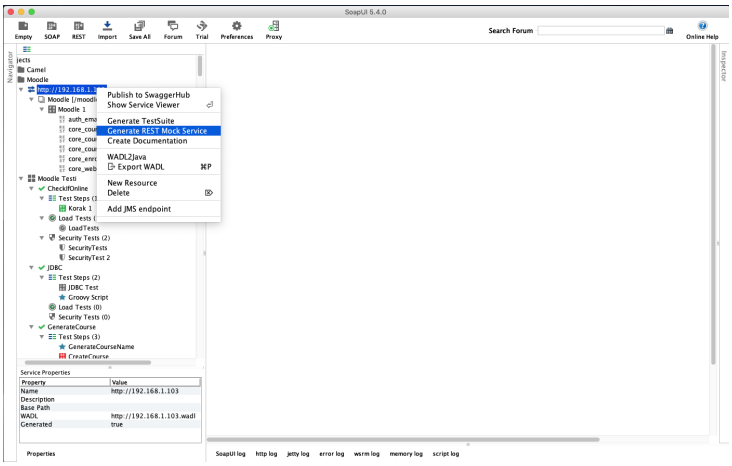
Slika 5.16: Dodajanje varnostnih testov



Slika 5.17: Dodani varnostni testi in okno za nastavitev beleženja

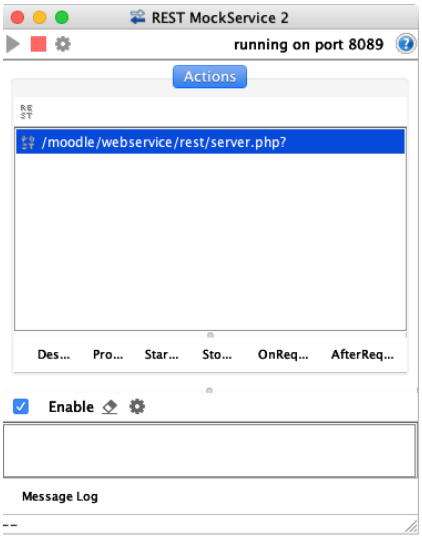
Nastavki servisov

Pri testiranju PO in objektno usmerjenem programiranju imenujemo z nastavki objekte, ki simulirajo obnašanje pravih objektov z namenom testiranja celotnega sistema, testiranja drugih objektov ali delov programske kode nastavki objektov (ang. mock object) [24].
Za nastavke objektov je značilno, da jih uporabljamo ko [7]:
- zahtevana storitev pri razvoju IS še ni na voljo,
- so izhodni podatki objekta, ki ga testiramo nedeterministični in je zato testiranje oteženo,
- je objekt počasen (komunikacija z oddaljeno bazo podatkov).
V ta namen orodje SoapUI uporabniku omogoča ustvarjanje nastavkov objektov (ang. mocking).
Uporabnik lahko ustvari nov nastavek neposredno iz REST ali SOAP zahtevka z izbiro opcije `Generate REST/SOAP Mock Service` (slika 5.18).



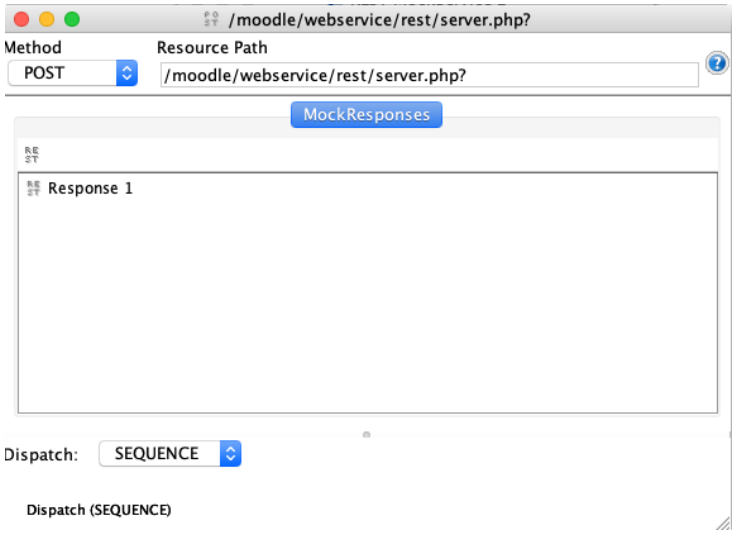
Slika 5.18: Ustvarjanje novega nastavka

Na sliki 5.19 vidimo nastavek med delovanjem.



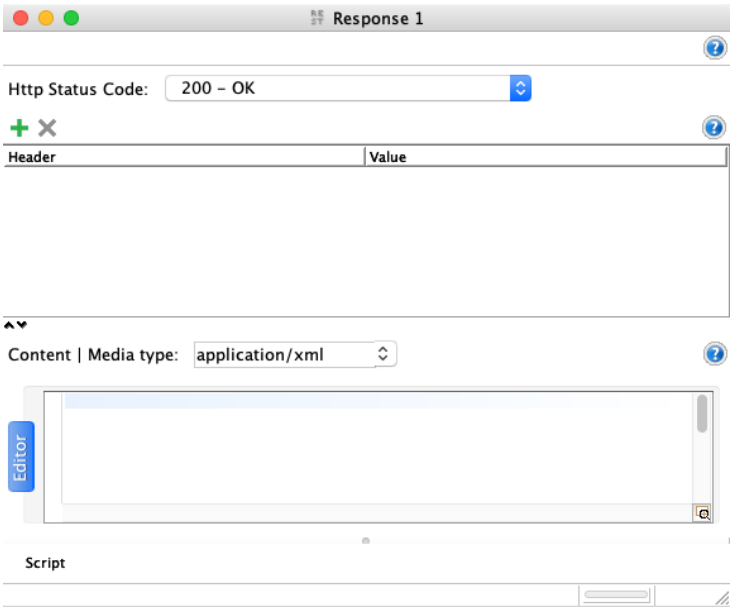
Slika 5.19: Okno nastavka

V drugem koraku uporabnika pričaka okno v katerem lahko zažene ali ustavi novo ustvarjen nastavek storitve. Kot privzeto se storitev zažene nemudoma ob kreiranju in posluša na vratih 8089. V tem oknu so vidne aktivnosti, ki jih lahko uporabnik dodaja ali ureja po potrebi. Obstajajo pa tudi možnosti dodajanja opisa, dodajanja lastnosti in dodajanja skript, ki se lahko izvedejo pred ali po izvedbi imitacije storitve imitacije storitve. Omeniti pa je potrebno še, da pri nastavkih lahko uporabnik definira tudi skripti, ki se zaženeta po in/ali pred izvedbo odgovora.



Slika 5.20: Okno odziva mock storitve

V oknu aktivnosti je možno pri nastavku REST storitve izbirati med metodami odziva: GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE ali PATCH. Uporabnik pa lahko kot odziv uporabi tudi Groovy skripto, ali pa lahko celo doda dodatne odzive (slika 5.20).



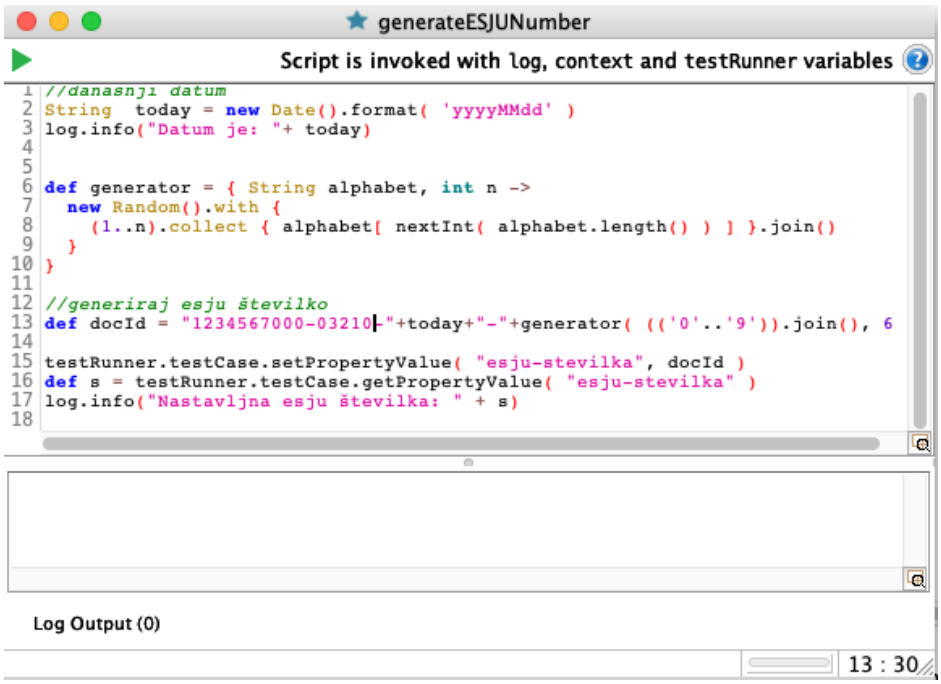
Slika 5.21: Nastavitveno okno odziva nastavka

V oknu za nastavitve odziva je možno nastaviti statusno HTTP kodo, dodati zaglavje, ki ga vrne odziv, polje `content - media type` in tudi poljubno Groovy skripto (slika 5.21).

Napredna avtomatizacija testov s pomočjo skriptiranja

Orodje SoapUI podpira uporabo programskih jezikov JavaScript in Groovy za skriptiranje (ang. scripting) (slika 5.22). Med razvojem testov se pogosto zgodi, da mora uporabnik sam ustvariti specifične vrednosti, ki se nato uporabijo v določenih testnih korakih ali nadaljnih testih oziroma se enostavno dodelijo kot vrednost spremenljivke znotraj istega projekta.
Skriptiranje se v orodju uporablja najpogosteje kot del testnega primera v obliki Groovy testnega koraka ali pa pred in po zagonu testnih primerov (ali nabora testnih primerov). Mogoče je določiti skripto, ki se zažene pred zagonom in skripto, ki se zažene po izvedbi testa. Uporabimo pa ga lahko tudi za razširjanje funkcionalnosti s pisanjem vtičnikov (ang. plugin) in ustvarjanje poljubnih validacijskih točk pri simulaciji storitve (tu se s pomočjo skripte

inicializira ali počisti stanje simuliranega servisa). Pri odpiranju in zapiranju projektov je tudi mogoče dodati skripto, ki se zažene pred odprtjem projekta in skripto, ki se zažene po zaprtju projekta.

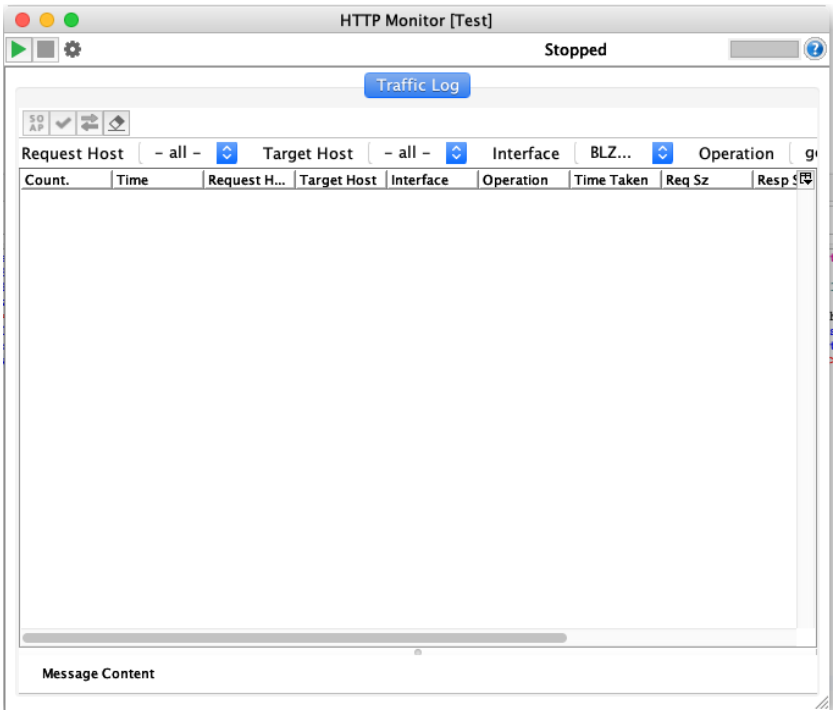


Slika 5.22: Groovy skripta kot testni korak

Vse skripte imajo dostop do več posebnih spremenljivk kot recimo `log` objekt preko katerega se lahko dodajajo dnevniški zapisi.

Opazovanje in prestrezanje HTTP/HTTPS sej

Orodje vsebuje implementacijo nadzornika (ang.proxy ali HTTP monitor) s katerim je mogoče prestrezati omrežni promet (slika 5.23).
Uporabniku omogoča shranjevanje, analizo in spreminjanje HTTP prometa s tehniko napada s posrednikom (ang. man in the middle). Prav tako je mogoče dešifrirati SSL promet, ko uporabnik poseduje pravilne ključe.



Slika 5.23: Okno vgrajenega posrednika

Nadzornik lahko deluje v dveh načinih. Posredniški način (ang. proxy), deluje kot klasični posrednik, tako da od odjemalca prejeta sporočila posreduje naprej do željenega spletnega naslova. Drugi način pa je HTTP tunel (ang. HTTP tunnel), ki ob pogoju da stranka komunicira direktno z orodjem SoapUI poskrbi, da posreduje njen zahtevek na ciljni spletni naslov. Možnost uporabe v načinu HTTP tunela je uporabna v primeru, da želi uporabnik dešifrirati SSL promet ali pa ob nezmožnosti uporabe posredniškega načina.

Analitika in poročila

V verziji SoapUI 5.4.0 (Community) ni mogoče generirati in izvažati poročil drugače kot v omejeni obliki po izvedbi obremenitvenih testov. Tu je možno izvoziti XML obliko poročila.

Čeprav ustvarjanje poročil v prosto dostopni verziji ni možno, pa je možno podrobno analizirati različne imeniške dnevnike, ki nastanejo med testira-

njem (ang. request log, test case log). Iz teh lahko uporabnik s pomočjo skriptnega programskega jezika Groovy sam oblikuje poročilo, kar so nekateri uporabniki tudi že storili [23].

Poglavje 6

Primer izvedbe testiranja z orodjem SoapUI

V tem poglavju predstavimo testiranje z orodjem SoapUI na kratkem praktičnem primeru, ki ga uporablja ekipa E2.

6.1 Analiza obstoječih testnih projektov

V namen razvoja podobnega primera s primernim naborom testov, za ekipo E2 smo analizirali šest SoapUI projektov in v okviru teh 30 naborov testnih primerov. Želeli smo poustvariti natančno strukturo naborov testov, kot jih uporablja ekipa. Podatek o povprečnem številu testnih korakov v testnem naboru smo potrebovali pri kreiranju lastnih testnih scenarijev in testnih naborov.

6.1.1 Povprečno število testnih korakov na posamezni nabor testov

Med analizo smo zaznali, da so nabori testov najpogosteje sestavljeni iz naslednjih števil testnih korakov: 7, 8, 9, 10, 13, 15, 17, 23.
Povprečno število testnih korakov na testni primer je tako 12,75 testnih ko-

rakov.

6.1.2 Različne strukture naborov testov

Pri tem smo opazili naslednje najpogostejše strukture testnih korakov:

1. ***primer:***
   HTTP zahtevek
   HTTP zahtevek
   SOAP zahtevek
   Premor(300ms)
   HTTP zahtevek
   JDBC zahtevek
2. ***primer:***
   HTTP zahtevek
   Groovy skripta
   SOAP zahtevek
   Premor(500ms)
   HTTP zahtevek
   JDBC zahtevek
   JDBC zahtevek
3. ***primer:***
   Groovy skripta
   HTTP zahtevek
   HTTP zahtevek
   SOAP zahtevek
   Premor(2000ms)
   SOAP zahtevek
   HTTP zahtevek
   HTTP zahtevek
   JDBC zahtevek

6.2 Opis testnega okolja

V namen prikaza testiranja z orodjem SoapUI smo vzpostavili testno okolje, ki je sestavljeno iz Mac OS X operacijskega sistema [6], virtualizacijske PO Oracle Virtualbox [32] in Ubuntu virtualnega strežnika [5]. Na strežnik smo namestili programsko opremo Moodle [16], ki nam omogoča več vrst testiranja in že privzeto vsebuje REST programski vmesnik.

6.2.1 Namestitev in konfiguracija testne aplikacije

Natančni opis namestitve in konfiguracije se nahaja v poglavju 9

6.3 Testni scenariji in razčlenitev na testne korake

Odločili smo se, da bomo testirali tri funkcionalnosti PO Moodle. V ta namen smo ustvarili naslednje testne scenarije.

1. **Ročno kreiranje novega uporabnika** ter **registracija novega uporabniškega računa.**
   Primer uporabe:
   1. Uporabnik ustvari nov uporabniški račun.
2. **Avtomatično kreiranje uporabniškega računa (uporabnika) s pomočjo integriranega API vmesnika PO Moodle.**
   Primer uporabe:
   1. Zunanji odjemalec s pomočjo REST klica ustvari novega uporabnika.
3. **Kreiranje novega tečaja s pomočjo integriranega API vmesnika PO Moodle.**
   Primer uporabe:
   1. Zunanji odjemalec s pomočjo REST klica ustvari nov tečaj.

6.3.1 Razčlenitev na testne korake

Na podlagi opisanih funcionalnosti (primerov uporabe) smo v nadaljevanju natančno razčlenili testne scenarije na testne korake.

Testni scenarij 1

Ime testnega primera: `Create user manually`
Testni koraki:

1. Ročno ustvari uporabnika.
2. Preveri v podatkovni bazi, če je bil uporabnik dejansko ustvarjen.

Pričakovan izid: Uspešno se ustvari nov uporabnik.

Testni scenarij 2

Ime testnega primera: `Create user automatically`
Testni koraki:

1. Generiraj potrebne uporabniške podatke.
2. Na podlagi generiranih uporabniških podatkov ustvari uporabnika.
3. Preveri v podatkovni bazi, če je bil uporabnik dejansko ustvarjen.

Pričakovan izid: Uspešno se ustvari nov uporabnik.

Testni scenarij 3

Ime testnega primera: `Create course automatically`
Testni koraki:

1. Generiraj potrebne podatke o tečaju.
2. Na podlagi generiranih podatkov ustvari nov tečaj.
3. Preveri v podatkovni bazi, če je bil nov tečaj uspešno ustvarjen.

Pričakovan izid: Uspešno se ustvari nov tečaj.

6.4 Identifikacija potrebnih API klicev

Pri implementaciji testnih korakov smo morali določiti funkcije API vmesnika, ki so potrebne za izvedbo opisanih testnih korakov. To sta funkciji [11]:

- `auth_email_signup_user`
  Obvezni argumenti funkcije so: `uporabniško ime (ang. username)`,

`geslo (ang. password)`, `ime (ang. firstname)`, `priimek (ang. lastname)` in `elektronski naslov (ang. email)`.

- `core_course_create_courses`
  Obvezni argumenti funkcije so: `polno ime (ang. fullname)`, `kratko ime (ang. shortname)`, `enolični določnik (ang. id)`.

Na podlagi funkcij smo ustvarili še dva REST zahtevka z zahtevanimi polji, na podlagi katerih smo ustvarjali potrebne testne korake.

6.5 Razvoj testnih primerov

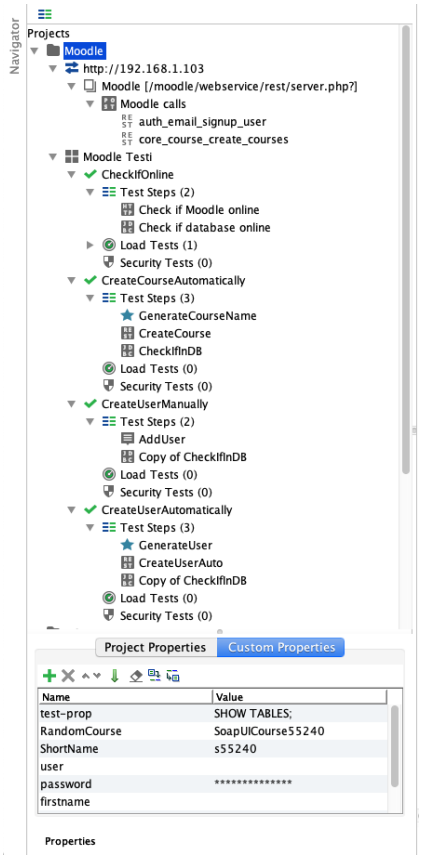
Razvoj testnih primerov se je pričel s kreiranjem novega projekta `Moodle` v katerem smo ustvarili prazen nabor testov, ki smo ga poimenovali `Moodle Testi`. Dodali smo osnovni REST zahtevek, kjer smo definirali IP naslov, nato (naslov virtualnega strežnika 192.168.1.103) smo razčlenili funkcionalnosti na testne korake in identificirali funkcije iz API vmesnika PO Moodle, ki jih bomo v ta namen potrebovali. Za gradnjo testnih primerov smo potrebovali naslednje tipe testnih korakov:

1. **Ročno kreiranje novega uporabnika**:
   Potrebovali smo testni korak tipa *ročni korak* in testni korak tipa *JDBC zahtevek*. S prvim smo opisali ročno akcijo testerja in kakšen naj bi bil pričakovan rezultat. Z drugim smo v podatkovni bazi preverili, če je bila akcija uspešno izvedena.

2. **Avtomatično kreiranje novega uporabnika**:
   Potrebovali smo testni korak *Groovy skripta*, *REST zahtevek* in *JDBC zahtevek*. Z Groovy skripto smo generirali potrebne spremenljivke (`up. ime`, `geslo`, `ime`, `priimek` in `elektronski naslov`), ter jih zapisali v lastnosti projekta. S korakom REST zahtevek smo izvedli klic, ki ustvari novega uporabnika. Na koncu smo s pomočjo JDBC zahtevka v podatkovni bazi preverili, če je bila akcija uspešna.

3. **Kreiranje novega tečaja**:
   Podobno kot pri avtomatičnem kreiranju novega uporabnika, *Groovy skripta* služi generiranju podatkov imena tečaja in kratkega imena tečaja. Koraka *REST zahtevek* in *JDBC zahtevek* sta imela enako vlogo kot zahtevka v prejšnjem primeru.

V izogib morebitnim napakam pri testiranju (recimo zaradi slabe povezave z virtualnim strežnikom ali nedosegljivosti podatkovne baze) smo dodatno ustvarili še dodatna testna primera, ki preverita dosegljivost strežnika in podatkovne baze. Izvedeta se pred izvedbo testov naštetih funkcionalnosti. Za to smo uporabili testna koraka *HTTP zahtevek* in *JDBC zahtevek*.



Slika 6.1: Ustvarjen projekt z nabori testnih primerov po zgoraj opisanih specifikacijah

Na sliki 6.1 vidimo opisan projekt z nabori testnih primerov.

6.5.1 Validacijske točke testnih korakov

Uspešnost izvedbe zgoraj opisanih testnih korakov smo preverjali z validacijskimi točkami.

- `Check if online` (dodatni testni primer)

Testni korak	Validacijska točka
`Check if Moodle online`	Vsebuje niz `ThesisMoodle`
`Check if database online`	Vsebuje niz `information_schema`

- `Generate Course`

Testni korak	Validacijska točka
`Generate course name`	/
`Create course`	Vsebuje niz VALUE
`Check if in DB`	Vsebuje spremenljivko `#Project#ShortName`

- `Create User Manually`

Testni korak	Validacijska točka
`Add user`	/
`Check if in DB`	Vsebuje nastavljeno uporabnisko ime `testuser1`

- `Create User Automatically`

Testni korak	Validacijska točka
`Generate user`	/
`Create user auto`	Vsebuje niz `This username already exists, choose another`
`Check if in DB`	Vsebuje spremenljivko `#Project#ShortUser`

6.6 Izvedba testiranj

Na koncu nam je ostala le še izvedba testiranja. Kot smo že opisali je zagon testov v orodju SoapUI enostaven, s pomočjo zelenega `play` gumba v levem gornjem kotu okna (slika 6.2).
Vsi testi so se uspešno izvedli že v prvem poskusu.

Slika 6.2: Okno za izvedbo testov zbirke testov `Moodle testi`

Poglavje 7

Predlogi za izboljšanje procesa testiranja in priporočila

Na podlagi ovrednotenja procesa razvoja PO v podjetju in praktične presoje uporabe testnega orodja SoapUI smo prišli do naslednjih ugotovitev:

- vloga testerja v ekipi ni ločena od ostalih vlog in nima dovolj avtonomije;
- ekipa ne upošteva določenih omejitev in priporočil metodologije SCRUM;
- uvajalni program je usmerjen samo v seznanitev novozaposlenega z interno politiko in strukturo podjetja;
- v podjetju ne obstaja noben mehanizem za seznanjanje zaposlenih z novimi tehnologijami;
- ekipa ne beleži rezultatov testiranj in prav tako ne vodi statistik o testiranju;
- orodje, ki ga uporablja ekipa za testiranje PO ne omogoča izvoza poročil v primernem formatu;

- orodje, ki ga uporablja ekipa za testiranje PO ni primerno za varnostna testiranja PO.

Zaradi naštetih ugotovitev predlagamo naslednja priporočila za izboljšanje procesa testiranja in razvoja PO:

1. **Ločitev vloge testerja in analitika ter uvedba ločene ekipe, ki se ukvarja izključno s testiranjem PO.**
   V podjetju se izvaja več projektov razvoja PO. Čeprav je uvajanje novih vlog oziroma delovnih mest zmeraj povezano z višjimi stroški, je v našem primeru pomembno iz več vidikov. Prvi je pridobitev avtonomije testerja, ki je nujna za objektivno izvedbo testiranja PO. Drugi razlog je razbremenitev glavnega razvijalca in s tem posledično izboljšanje kvalitete programske kode. Posebna ekipa bi tako delovala in se ukvarjala izključno s testiranjem PO in izvajala testiranje na več projektih v podjetju.

2. **Upoštevanje omejitve trajanja dnevnih sestankov.**
   Dnevni sestanki so po metodologiji SCRUM strogo omejeni na 15 minut. V skupini se redno dogaja, da dnevni sestanki trajajo tudi preko 40 minut. Velikokrat se med arhitektom in glavnim razvijalcem razvijejo trenja, ki jih rešujeta znotraj dnevnega sestanka. Ostali člani ekipe s tem izgubijo dragocen čas, ki bi ga sicer namenili razvoju. Priporočamo, da razvojna ekipa upošteva omejitev 15 minut.

3. **Uvedba uvajalnega programa za novozaposlene projektne vodje in testerje.**
   V podjetju že sedaj poteka mentorski program za novozaposlene, ki pa se ne izvaja temeljito oziroma je usmerjen predvsem v seznanjanje s pravili podjetja in organizacijsko strukturo. V okviru tega programa mora novozaposleni opraviti izpit v roku šestih mesecev, ki ga opravlja pred upravo podjetja. Izpit je usmerjen v poznavanje organizacijske strukture podjetja. Predlagamo, da se uvedejo strokovni mentorski programi, s pomočjo katerih se bodo novozaposleni seznanili s tehničnimi

značilnostmi projekta in procesom razvoja in testiranja PO.

4. **Mehanizem za upravljanje in podporo vpeljavi novih tehnologij.**
   Čeprav se zavedamo, da so podjetja velikorat prisiljena uporabljati zastarele tehnologije bodisi zaradi zahtev strank, zaradi pomanjkanja sredstev ali časa pa je pomembno, da se sčasoma vpeljujejo nove tehnologije. To je pomembno predvsem iz vidika informacijske varnosti. Predlagamo, da se v podjetju uvede mehanizem postopnega vpeljevanja novih tehnologij in obenem tudi mehanizem seznanjanja kadra z novimi tehnologijami.

5. **Beleženje statistik testiranj.**
   Ob sodelovanju v vseh fazah procesa razvoja in testiranja PO smo ugotovili, da se ne vodi nobenih statistik ali poročil o testiranju. Beleženje je nujno potrebno, saj na podlagi le-tega lahko odpravimo morebitne sistematične napake, ki jih drugače ne bi opazili.

6. **Formalni pregled testnih primerov.**
   Testne primere trenutno razvija in pregleduje predvsem glavni razvijalec (starejši razvijalec 1), kar pomeni, da obstaja določena mera pristranskosti. V primeru, da bi podjetje imelo ločeno skupino za izvedbo testiranja PO, bi lahko omenjena skupina izvedla tudi preglede testnih primerov.

7. **Sledenje napakam med procesom razvoja PO in beleženje.**
   Beleženje napak med testiranjem ni dovolj, saj obstaja nevarnost, da take napake ostanejo zabeležene in se jih ne odpravi, če le-te niso kritične. To lahko vodi v zelo slabo prakso, katere posledice se lahko odražajo kot potencialne ranljivosti. Ker se v podjetju že sedaj uporablja PO Jira [30], predlagamo da testerji sproti vnašajo odkrite napake med procesom testiranja neposredno v Jiro. S tem bo ekipa dosegla sledljivost statusom odkritih napak in jih tako učinkovito sproti tudi odpravljala.

8. **Vpeljava ločenih orodij za varnostne preglede programske kode.**
   Varnostna preverjanja programske kode morajo biti izvedena v večini primerov s strani nepristranskega ocenjevalca. To pomeni, da take vrste preverjanj izvaja nekdo, ki ni sodeloval pri razvoju PO. To seveda ne pomeni, da se tovrstno testiranje ne izvede pred uporabniškim testiranjem. Ker se je včasih pojavila potreba po varnostnih testiranjih še pred predajo naročniku smo opazili, da SoapUI ni dovolj zmogljiv za taka testiranja. Predlagamo, da se v proces testiranja uvede namenska PO za testiranje varnosti in statično analizo programske kode (recimo orodje Checkmarx [31]).

9. **Uporaba orodja za testiranje z možnostjo izvoza razširjenih poročil.**
   Odprtokodna različica orodja SoapUI omogoča izvoz poročil v omejeni obliki in izključno v formatu XML. Za namene natančne analize procesa testiranja potrebuje ekipa vpogled v podatke o testiranjih, ki so že bila izvedena. Za strukturirano hrambo tovrstnih podatkov mora imeti ekipa način, da izvozi in shrani podrobna poročila o testiranjih v poljubnih formatih. To omogoča plačjiva verzija orodja SoapUI Pro [29].

S temi ukrepi bi nivo procesa testiranja programske opreme uspeli približati drugi stopnji modela stopenj zrelosti CMM.

Poglavje 8

Sklepne ugotovitve

V sklopu diplomske naloge smo predstavili ozadje in delovanje razvojne ekipe E2. S ciljem, da ju objektivno ovrednotimo na podlagi ustreznega postopka. Osredotočili smo se na proces razvoja in testiranja PO. Na podlagi analize procesa smo s pomočjo modela CMM ocenili, da se podjetje nahaja šele na prvi oziroma začetni stopnji. V nadaljevanju smo predstavili odprtokodno orodje SoapUI s katerim razvojna ekipa testira PO. Analizirali smo nekaj realnih naborov testov in na podlagi rezultatov prikazali običajni potek testiranja. Oboje nam je služilo pri presoji, kako izboljšati izvedbo testiranja v podjetju. V ta namen smo predlagali več konkretnih priporočil za izboljšanje procesa testiranja PO.

Ugotovili smo, da je ločitev funkcije testerja in razvijalca ključna pri objektivnosti rezultatov testiranja PO. Dnevni sestanki pa naj se izvajajo znotraj časovnega okvirja petnajstih minut. Pomemben dejavnik pri uspešnosti novozaposlenih je tudi kvaliteta in strokovnost uvajalnega programa, še zlasti, ko je le-ta vključen v razvoj in testiranje PO. Prav tako gre poudariti, da podjetje mora, če želi razvijati kvalitetno PO, ki je v koraku z modernimi smernicami razvoja PO redno seznanjati zaposlene z novimi tehnologijami. Med analizo procesa razvoja in testiranja PO se je izkazalo, da se ne beleži napak med testiranjem, kakor tudi dejstvo, da se ne zbira nobenih statistik o procesu testiranja. S tem je ekipa prikrajšana za dragocene podatke, ki bi

jih lahko iz statistik pridobila in na podlagi teh izboljšala proces razvoja in testiranja PO. Zaradi časovnih in tudi ostalih omejitev smo se v diplomski nalogi osredotočili na proces testiranja PO in ga ovrednotili samo na podlagi modela CMM. Vsekakor pa bi lahko še:

- proces razvoja in testiranja PO ocenili s pomočjo modelov kot recimo CMMI [8] in TMMI [10]. S pomočjo modela PCMM [9] bi lahko denimo ovrednotili nivoja upravljanja podjetja in upravljanja s človeškimi viri, ter nato raziskali vpliv kvalitete upravljanja s človeškimi viri na proces testiranja PO.

- odprtokodno orodje SoapUI in sami implementirali funkcionalnost izvoza in kreiranja razširjenih poročil. To bi lahko storili na način neposredno implementacije v orodje ali pa s pomočjo skript razvitih v programskem jeziku Python [28], ki bi na podlagi zajetih dnevniških zapisov izvedenih testov generirala podrobna poročila.

- preučili možnost popolnoma avtomatične gradnje testnih naborov s pomočjo orodja SoapUI.

Upamo, da bodo rezultati in ugotovitve diplomskega dela pomagala podjetjem pri izboljšavi procesa razvoja in testiranja PO ter tako pripomogla k razvoju kvalitetnejše PO.

Poglavje 9

Priloge

V tem poglavju se nahaja opis namestitve testne aplikacije in izvorna koda testnih primerov.

9.1 Verzije programske opreme

Operacijski sistem: `Mac OS 10.14.2`
Virtualizacija: `Oracle VirtualBox 6.0`
Virtualni strežnik: `Ubuntu 16.04`
Testirana aplikacija: `Moodle: 3.3.9`

9.2 Namestitev testnega sistema

Programsko opremo Moodle smo namestili z izvedbo naslednjih korakov [37]:

```
1. sudo add-apt-repository ppa:ondrej/php
2. sudo add-get update
3. sudo apt-get install apache2
mysql-client mysql-server php7.0 libapache2-mod-php7.0
4. sudo apt-get install
graphviz aspell ghostscript
```

```
clamav php7.0-pspell php7.0-curl
php7.0-gd php7.0-intl php7.0-mysql php7.0-xml php7.0-xmlrpc
php7.0-ldap php7.0-zip
php7.0-soap php7.0-mbstring
5. sudo service apache2 restart
6. sudo apt-get install git-core
7. sudo git branch -a
8. sudo git branch --track MOODLE_33_STABLE originMOODLE_33_STABLE
9. sudo git checkout MOODLE_33_STABLE
10. sudo cp -R /opt/moodle /var/www/html/
11. sudo mkdir /var/moodledata
12. sudo chown -R www-data /var/moodledata
13. sudo chmod -R 777 /var/moodledata
14. sudo chmod -R 0755 /var/www/html/moodle
15. sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf
```

Spremembe v `mysqld.conf` datoteki:

```
default_storage_engine = innodb
innodb_file_per_table = 1
innodb_file_format = Barracuda
```

Spremembe in dodatni ukazi v `mysql ukazni lupini`:

```
CREATE DATABASE moodle DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
create user 'moodle'@'localhost' IDENTIFIED BY 'moodle600';
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,CREATE TEMPORARY TABLES,DROP,
INDEX,ALTER ON
moodle.* TO moodledude@localhost IDENTIFIED BY 'moodle600';
```

9.3 Programska koda testnih primerov

9.3.1 Programska koda skripte za generiranje podatkov novega tečaja

```
String courseName = "SoapUICourse"
int random = (int)(6667 + 256*Math.random())
String sh = "s";
testRunner.testCase.testSuite.
project.setPropertyValue("RandomCourse",courseName+random)
testRunner.testCase.testSuite.
project.setPropertyValue("ShortName",sh+random)
```

9.3.2 Programska koda skripte za generiranje podatkov novega uporabnika

```
int random = (int)(6667 + 256*Math.random())
String usr = "username"
String pass = "Password"
String email = "@thesisemail.com"
String firstname = "John"
String lastname = "Doe"

testRunner.testCase.
testSuite.setPropertyValue("user",username+random)
testRunner.testCase.
testSuite.setPropertyValue("password",pass+random+"!")
testRunner.testCase.
testSuite.setPropertyValue("email",username+random+email)
testRunner.testCase.
testSuite.setPropertyValue("firstname",firstname)
testRunner.testCase.
```

```
testSuite.setPropertyValue("lastname",lastname)
```

Literatura

[1] EUPL licenca v1.2 . https://opensource.org/licenses/EUPL-1.2. Dostopano: 06.02.2019.

[2] Heroic programming. http://wiki.c2.com/?HeroicProgramming. Dostopano: 17.03.2019.

[3] HL7 Standards. http://www.hl7.org/implement/standards/product_section.cfm?section=1. Dostopano: 06.02.2019.

[4] Introduction to XML. https://www.ibm.com/developerworks/xml/tutorials/xmlintro/xmlintro.html. Dostopano: 17.03.2019.

[5] Linux distribucija Ubuntu(Server). https://www.ubuntu.com/server. Dostopano: 17.03.2019.

[6] Mac OS. https://www.apple.com/si/macos/mojave/. Dostopano: 17.03.2019.

[7] Mock object, Wikipedia. https://en.wikipedia.org/wiki/Mock_object. Dostopano: 06.02.2019.

[8] Model CMMI 2.0. https://cmmiinstitute.com/. Dostopano: 17.03.2019.

[9] Model PCMM. https://cmmiinstitute.com/pm. Dostopano: 17.03.2019.

[10] Model TMMI. https://www.tmmi.org/tmmi-model/. Dostopano: 17.03.2019.

[11] Moodle, Creating a web service client. https://docs.moodle.org/dev/Creating_a_web_service_client. Dostopano: 06.02.2019.

[12] OASIS Standards. https://www.oasis-open.org/standards#wssv1.1.1=OASIS. Dostopano: 06.02.2019.

[13] OTA Bitmap . http://fileformats.archiveteam.org/wiki/OTA_bitmap. Dostopano: 06.02.2019.

[14] Portal e-uprava. https://e-uprava.gov.si. Dostopano: 25.01.2019.

[15] Programska knjižnica Swing. https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html. Dostopano: 17.03.2019.

[16] Programska oprema Moodle. https://moodle.org/. Dostopano: 01.02.2019.

[17] Protokol SOAP, w3org. https://www.w3.org/TR/soap12-part1/#intro. Dostopano: 06.02.2019.

[18] Representational State Transfer (REST). https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. Dostopano: 17.03.2019.

[19] RSS 2.0 Specification. http://www.rssboard.org/rss-specification. Dostopano: 06.02.2019.

[20] Scrum Guide. https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf. Dostopano: 06.02.2019.

[21] SOAP Message Transmission Optimization Mechanism. https://www.w3.org/TR/soap12-mtom/. Dostopano: 06.02.2019.

[22] Spletna stran s povzetkom tehnike jadrnice. https://luis-goncalves.com/sailboat-exercise-sailboat-retrospective/. Dostopano: 17.03.2019.

[23] StackOverflow diskusija. https://stackoverflow.com/questions/41700437/creating-a-test-report-from-project-level-tear-down-script/41759553#41759553. Dostopano: 06.02.2019.

[24] Stubs and mocks, Microsoft. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff798400(v=pandp.10). Dostopano: 06.02.2019.

[25] SVG 1.1 (Second Edition). https://www.w3.org/TR/SVG11/. Dostopano: 06.02.2019.

[26] Uradna dokumentacija programske opreme SoapUI. https://www.soapui.org/soapui-projects/soapui-projects.html. Dostopano: 25.01.2019.

[27] Uradna spletna stran inštituta Software Engineering Institute - Carnegie Mellon University. https://www.sei.cmu.edu/. Dostopano: 25.01.2019.

[28] Uradna spletna stran jezika Python. https://www.python.org/. Dostopano: 17.03.2019.

[29] Uradna spletna stran orodja SoapUI Pro. https://www.soapui.org/professional/soapui-pro.html. Dostopano: 17.03.2019.

[30] Uradna spletna stran programske opreme Atlassian Jira. https://www.atlassian.com/software/jira. Dostopano: 25.01.2019.

[31] Uradna spletna stran programske opreme Checkmarx. https://www.checkmarx.com/. Dostopano: 16.03.2019.

[32] Uradna spletna stran programske opreme Oracle Virtualbox. https://www.virtualbox.org/. Dostopano: 17.03.2019.

[33] Uradna spletna stran programske opreme SoapUI. https://www.soapui.org/. Dostopano: 25.01.2019.

[34] Uradna spletna stran programskega jezika Apache Groovy. http://groovy-lang.org/. Dostopano: 06.02.2019.

[35] Uradna spletna stran programskega jezika Oracle Java. https://www.oracle.com/java/. Dostopano: 17.03.2019.

[36] Uradni vodič programske opreme SoapUI. https://www.soapui.org/getting-started.html. Dostopano: 25.01.2019.

[37] Uradni vodič za namestitev Moodle programske opreme. https://docs.moodle.org/36/en/Step-by-step_Installation_Guide_for_Ubuntu. Dostopano: 06.02.2019.

[38] What is Scrumban? https://www.agilealliance.org/what-is-scrumban/. Dostopano: 06.02.2019.

[39] What is vertical communication? Advantages and disadvantages of vertical communications. https://thebusinesscommunication.com/vertical-communication-advantages-and-disadvantages/. Dostopano: 06.02.2019.

[40] WS-Addressing Submission Request to W3C. https://www.w3.org/Submission/2004/05/. Dostopano: 06.02.2019.

[41] WSDL disclosure. http://www.ws-attacks.org/WSDL_Disclosure. Dostopano: 06.02.2019.

[42] WSDL, w3org. https://www.w3.org/TR/wsdl20. Dostopano: 06.02.2019.

[43] XHTML 1.0 (Second Edition). https://www.w3.org/TR/xhtml1/. Dostopano: 06.02.2019.

[44] XML Essentials. https://www.w3.org/standards/xml/core. Dostopano: 17.03.2019.

[45] Zapis James Strachana na svojem blogu. http://radio.weblogs.com/0112098/2003/08/29.html. Dostopano: 06.02.2019.

[46] Kent Beck and Erich Gamma. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[47] Alistair Cockburn. *The Costs and Benefits of Pair Programming*. 2000.

[48] Mike Cohn. *Agile estimating and planning*. Pearson Education, 2005.

[49] Alan Cooper et al. *The inmates are running the asylum:[Why high-tech products drive us crazy and how to restore the sanity]*. Sams Indianapolis, 2004.

[50] David Gelperin and Bill Hetzel. The growth of software testing. *Communications of the ACM*, 31(6):687–695, 1988.

[51] Watts S Humphrey and WL Sweet. A method for assessing the software engineering capability of contractors: preliminary version. Technical report, Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 1987.

[52] Henrik Kniberg and Mattias Skarin. *Kanban and Scrum-making the most of both*. Lulu. com, 2010.

[53] Glenford J Myers, Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.

[54] Eric Newcomer. *Understanding Web Services: XML, Wsdl, Soap, and UDDI*. Addison-Wesley Professional, 2002.

[55] Walker Royce. Cmm vs. cmmi: From conventional to modern software management. *The Rational Edge*, 2, 2002.

[56] Igor Rožanc and Viljan Mahnič. Uporaba modela CMM v majhnih organizacijah za razvoj programske opreme. *Elektrotehniski vestnik*, 70(3):149–154, 2003.